Nanotechnology Perceptions
ISSN 1660-6795
www.nano-ntp.com

A Novel Data Locality-Aware Scheduler for Improved Cloud Performance

Tapankumar A. Kakani[1], Rajkumar Muthusamy[2], Anuteja Reddy Neravetla[3], Karim Hudani[4], Ketan Gupta[5], Nasmin Jiwani[5]

[1]Software Developer, Pactiv Evergreen Inc, Mundelein, IL, USA
[2]Software Development Specialist Advisor, NTT Data, Irving, Texas, USA
[3]Department of Information Technology, University of the Cumberlands, Kentucky, USA
[4]Independent Researcher, Tampa, FL, USA
[5]Research Scientist, Dept. of Information Technology, University of The Cumberlands, Williamsburg, KY, USA

Scheduling tasks closer to stored data can significantly reduce network traffic. By optimizing for data locality, tasks can be matched with their associated data on the same node, minimizing the need for data transfer. However, many existing schedulers overlook the balance between task placement, data transfer overhead, and bandwidth consumption, focusing only on locality. We present a novel Genetic Algorithm-based Data Locality Scheduler (GADLS), which aims to balance time consumption and network bandwidth while improving data locality and throughput. GADLS employs a genetic algorithm to model data-task placement as a chromosome, optimizing for configurations that maximize locality and minimize bandwidth use. It integrates a multi-objective fitness function, balancing data movement, network traffic, and task runtime, with adaptive mutation and crossover mechanisms to explore a broad range of placement options. Through this approach, GADLS achieves an improvement of 18% in data locality rate and a 27% increase in throughput, demonstrating its effectiveness in maximizing resource utilization and enhancing performance in distributed environments.
Keywords: data locality, multi-tenancy, scheduling, Genetic Algorithm-based Data Locality Scheduler, cloud computing.

1. Introduction

In today's world of distributed computing, Wireless Sensor Networks (WSNs) and cloud-

based systems play a crucial role in various applications, including environmental monitoring, industrial automation, and real-time data processing[1]. A significant challenge these systems face is the need to minimize network traffic, which directly affects both performance and efficiency[2]. The necessity to transfer data between remote nodes often exacerbates network traffic, leading to higher latency, increased energy consumption, and diminished throughput[3]. One effective approach to address this issue is through optimizing data locality. This involves scheduling tasks and their related data on the same node, thereby reducing the need for inter-node data transfers. By optimizing data locality, we can significantly decrease network bandwidth usage and improve the overall performance of distributed systems. However, achieving optimal data locality is not straightforward, particularly when trying to balance other important factors like task runtime and network bandwidth.Although current scheduling algorithms aim to enhance data locality, they frequently overlook the trade-offs between task placement, data transfer costs, and bandwidth usage. Many traditional schedulers operate under the assumption that merely placing tasks and data in close proximity will yield optimal performance, without taking into account the additional expenses associated with data transfers or the effects on overall system throughput. This limited perspective results in inefficiencies in practical scenarios, where multiple objectives must be balanced to attain the best performance. Additionally, most existing scheduling methods lack the adaptability needed to manage dynamic, large-scale distributed environments where task demands and resource availability are in constant flux.

To overcome these challenges, we present the Genetic Algorithm-based Data Locality Scheduler (GADLS), a new method aimed at improving data locality while also balancing network bandwidth and time efficiency. The main goals of GADLS are: Minimize data transfer: By positioning tasks and their related data on the same node, GADLS greatly decreases the need for communication between nodes, which in turn reduces network traffic. Balance time consumption and bandwidth usage in GADLS aims to find an optimal balance between where tasks are placed and the data transfer overhead, ensuring the system runs efficiently without straining the network. Enhance throughput and resource utilization: The scheduler boosts system throughput by maximizing the use of resources while ensuring fair distribution of resources among competing tasks.

The contributions of this research include the proposed method employs a genetic algorithm to represent task-data placement as chromosomes, allowing for intelligent exploration of scheduling options. Multi-Objective Optimization: GADLS features a fitness function that takes into account both data locality and network resource usage, creating a balanced optimization framework for task scheduling. Adaptive Genetic Operations: The use of adaptive mutation and crossover techniques ensures effective exploration of the solution space, helping to avoid local optima and achieve improved scheduling results. Performance Improvements: Experimental findings show that GADLS enhances data locality by 18% and throughput by 27%, confirming its effectiveness in practical distributed systems.

2. Related Work

Task scheduling and data placement in distributed systems have become a significant focus

due to their impact on performance and resource utilization. Optimizing data locality is essential for reducing network overhead, yet traditional schedulers like Hadoop's FIFO often prioritize fairness over locality. Recent studies, such as Ali et al. [4], have proposed locality-aware scheduling to minimize remote data fetches, although there are still throughput trade-offs when handling heavy workloads. Genetic Algorithms (GAs) are commonly employed to tackle NP-hard task allocation issues, as shown by Kumar et al. [5], who highlighted their effectiveness in balancing workloads and optimizing resource usage, but their application in data placement strategies is still limited. Multi-objective resource allocation methods, including Dominant Resource Fairness (DRF), promote fair distribution and enhanced throughput, as noted by Zhang et al. [6], yet their use in dynamic environments requires further investigation. Moreover, bipartite graph-based models have proven useful for task-node mapping; Liu et al. [7] applied weighted graphs to improve data locality and load balancing, although these approaches often struggle to adapt to changing workloads. Machine learning methods, such as Multilayer Perceptrons (MLPs), have demonstrated potential in predicting task runtimes and resource release times, leading to more informed scheduling choices, as observed by Wang et al. [8], but their computational demands can be a drawback. Together, these foundational studies motivate the creation of the Genetic Algorithm-Based Data Locality Scheduler (GADLS), which combines genetic algorithms, bipartite graph modeling, and fairness principles to enhance task scheduling and optimize data placement in distributed systems.

Data locality-aware scheduling has emerged as a vital research focus in distributed systems and cloud computing, with numerous studies aimed at enhancing task execution efficiency by taking into account both data and task placement. Choi et al. [9] worked on improving data locality within Hadoop's MapReduce framework through a technique known as delay scheduling. This method allows tasks to be scheduled when the necessary data blocks are available on local nodes, thereby reducing network traffic and boosting data locality. Choudhury et al. [10] explored energy-efficient scheduling strategies aimed at optimizing data locality. Their study highlighted that by balancing task scheduling with energy constraints, it is possible to reduce power consumption while maintaining a high level of data locality. They suggested techniques that adaptively modify scheduling policies based on system load and task dependencies. In the realm of task scheduling and resource allocation, Qiu et al. [11] introduced a method that specifically enhances data locality by minimizing communication overhead between tasks and their associated data. Their approach ensures that tasks are assigned to nodes where their corresponding data resides, thus improving performance by avoiding the need for remote data access. Wang et al. [12] presented dynamic data placement strategies designed for large-scale distributed systems. Their approach modifies data placement during execution to ensure that data is allocated in a manner that reduces network congestion while maximizing task efficiency. This dynamic strategy is especially crucial in systems experiencing variable workloads.

Zhang et al. [13] put forward a fair scheduling algorithm that takes into account both data locality and resource allocation. Their algorithm ensures that tasks are assigned to nodes in a way that optimizes both computation and data access, thereby enhancing the overall throughput of the system. They contend that fairness in scheduling can lead to improved resource utilization and greater data locality. Sharma, Gupta, and Choudhury [14] proposed

a hybrid genetic algorithm aimed at enhancing data locality and task scheduling within cloud systems. Their method emphasizes optimizing task placement to reduce data transfer overhead while maximizing resource utilization. The study showcases the algorithm's ability to balance computational loads and improve system throughput in cloud environments.

3. Methodology and Implementation

The proposed Genetic Algorithm-based Data Locality Scheduler (GADLS) aims to enhance task scheduling by focusing on data locality and resource efficiency. It utilizes a bipartite graph matching framework that reflects the distribution of data blocks alongside the computational performance of the nodes. A job is deemed complete only when all its sub-tasks have been executed. To facilitate this, nodes are prioritized based on their computational idleness and utilization, with tasks assigned first to those with higher priority. The node that demonstrates the highest available resource utilization is chosen as the main candidate for task processing.

Bipartite Graph Model for Task Scheduling

The problem of data placement is formulated using a weighted bipartite graph $G = (T \cup S, E)$, where T represents the Set of tasks, S represents the Set of nodes and E⊆T×S defines the Set of edges connecting tasks and nodes. The weight wei(t,i) on edge e(t,i) denotes the available resource utilization of node n_i for task t. The resource utilization of node n_i is given as $u_{res}(n_i)$, and the available resource utilization is computed as:

$$u_{\text{ava\_res}}(n_i) = 1 - u_{\text{res}}(n_i) \quad (1)$$

If a task t has multiple potential nodes, it selects the node n_s with the highest weight wei(t,s), ensuring efficient resource allocation and reduced remote data access. The allocation function f:T→S maps each task t to a noden_f(t), aiming to minimize task execution delays while maintaining balanced load distribution.

Task Allocation Using Genetic Algorithm

The task allocation is treated as a maximum weighted bipartite matching problem, where the objective is to find an optimal mapping that minimizes remote task execution and improves throughput. The Genetic Algorithm (GA) is employed to solve this optimization problem effectively.

1. Encoding: Represent each task-to-node allocation as a chromosome. Each task-to-node allocation is represented as a chromosome. A chromosome encodes the mapping of tasks T to nodes S, where each task ti ∈ T is assigned to a node nj ∈ S. A solution (chromosome) consists of a series of task-node assignments:

$$\text{Chromosome} = \{(t1, ni1), (t2, ni2), \ldots, (tk, nik)\} \quad (3)$$

where t1 is the task and nik is the assigned node for task ti.

2. Fitness Evaluation: Calculate the fitness based on data locality, resource utilization, and minimized data transfer. The fitness of each solution is evaluated based on data locality and

resource utilization. We aim to maximize the fitness function F that balances these two objectives:

$$F = \alpha \times \text{Data Locality} + \beta \times \text{Resource Utilization} \quad (4)$$

3. Selection: Choose parent chromosomes based on their fitness scores. The fitness of each chromosome is evaluated based on two main factors: data locality and resource utilization. We aim to maximize the fitness function F, which is a weighted sum of these two objectives.

$$F = w1 \times \text{Data Locality Score} - w2 \times \text{Resource Utilization Score} \quad (5)$$

4. Crossover and Mutation: Generate new task allocations by combining and mutating parent solutions to explore diverse allocations.Example of crossover between two chromosomes C1 and C2:

$$C1 = \{(t1, n1), (t2, n2), \dots\}, C2 = \{(t1, n3), (t2, n4), \dots\} \quad (6)$$

After crossover, the resulting offspring might be:

$$C3 = \{(t1, n1), (t2, n4), \dots\}, C4 = \{(t1, n3), (t2, n2), \dots\} \quad (7)$$

Mutation introduces randomness by modifying one or more elements of the chromosome. For example, task ti assigned to node nj may be randomly reassigned to another node:

$$\text{Mutation: } (ti, nj) \rightarrow (ti, nk) \quad (8)$$

This allows the algorithm to explore new task-node allocations that may not have been considered during crossover.

5. Termination: Stop when an optimal or near-optimal allocation is found.

Data Transfer Time Estimation

The decision to move data or computation depends on estimating the data transfer time $T\text{tran}(t)$ and resource release time $W_{\text{rel}}(\alpha(t))$. The data transfer time is calculated using:

$$[T_{\text{tran}}(t) = \frac{|dt| \times \text{rtt}(n_i, n_j)}{\text{twt}}] \quad (2)$$

Where, $|\,dt\,|$ represents the Size of the data block, $\text{rtt}(n_i n_j,)$ represents the Round-trip time between nodes n_i and $n_j$and twt represents the TCP window size. To estimate the remaining execution time of tasks, a Multi-Layer Perceptron (MLP) model is incorporated into GADLS. This MLP model, built with Keras, delivers accurate predictions for resource release time, which aids in optimizing task placement decisions. These predictions enhance the scheduler's efficiency by allowing it to adapt dynamically to changes in workload.

The proposed research aimed at enhancing task scheduling and data placement in distributed systems. It starts by creating a weighted bipartite graph $G = (T \cup S, E)$, where tasks (T) symbolize computational jobs, nodes (S) represent computational resources, and edges (E) illustrate the relationships between tasks and nodes, with weights determined by the utilization of available resources $u_{\text{ava\_res}}(n_i) = 1 - u_{\text{res}}(n_i)$. Chromosomes in this

framework encode the mappings of tasks to nodes, reflecting potential scheduling solutions. An initial population is generated randomly to maintain diversity within the solution pool. Each chromosome undergoes evaluation through a fitness function that takes into account data locality, bandwidth usage, and task runtime, ensuring that solutions focus on minimizing data transfer and optimizing execution. The fitness function integrates edge weights and calculates data transfer time as equation (2), which is used to rank the chromosomes.

Efficient task scheduling and resource allocation present significant challenges in distributed systems, particularly in settings like cloud computing and wireless sensor networks. The Genetic Algorithm-based Data Locality Scheduler (GADLS) has been developed to tackle these issues by utilizing genetic algorithms to optimize where tasks are placed and enhance data locality. By reducing data transfer overhead and ensuring balanced resource use, GADLS improves both system throughput and energy efficiency. This algorithm employs sophisticated genetic operations, including tournament selection, crossover, and adaptive mutation, to effectively navigate the solution space. Additionally, it uses a maximum weight matching strategy to assign tasks to nodes based on their available resources and preferences for data locality. To promote fairness and mitigate resource contention, Dominant Resource Fairness (DRF) is implemented, ensuring that resources are distributed equitably among tasks and tenants. Through its iterative methodology, GADLS refines task scheduling and resource allocation, showing notable enhancements in performance metrics such as throughput, data locality, and energy efficiency. The following steps detail how GADLS operates to achieve these goals.

The algorithm starts by generating an initial population of N chromosomes. Each chromosome corresponds to a distinct mapping of tasks to nodes. This random initialization promotes diversity within the solution space, which is crucial for exploring different potential task-node mappings and preventing premature convergence. Each chromosome is evaluated based on data locality, bandwidth usage, and task runtime. The fitness function ensures solutions prioritize task placement that minimizes data transfer and improves execution efficiency. Tasks are assigned to nodes based on the highest edge weight wei(t,s), ensuring optimal placement. A judgment mechanism decides task reassignment or waiting when resource constraints arise.Dominant Resource Fairness ensures equitable resource distribution among tenants, prioritizing dominant resource demands while maintaining fairness and system throughput.

Algorithm 1: GADLS Scheduler

1: **Input:** Task set T, node set S, resource utilization u_{res}, and data locality information.

2: **Output:** Optimized task-node mapping with improved data locality and system throughput.

Initialization

3: Construct a weighted bipartite graph $G = (T \cup S, E)$.

4: $wei(t, s)$: Edge weights representing the available resource utilization $u_{ava\_res} = 1 - u_{res}$.

Task Scheduling Process

5: **while** (tasks t remain unscheduled in T) **do**

6: **for** (each task t in T) **do**

7: // Identify candidate nodes for t.

8: **for** (each node s in S) **do**

9: **if** (node s contains data required by t) **then**

10: // Add node to candidate set.

11: $S \leftarrow S \cup s$.

12: // Calculate edge weight based on resource availability.

13: $wei(t, s) \leftarrow u_{ava\_res}(s)$.

14: $E \leftarrow E \cup (t, s)$.

15: **end if**

16: **end for**

Maximum Weight Matching

17: Find feasible node $n_{f(t)}$ for task t with:

18: $n_{f(t)} \leftarrow \text{argmax}_{s \in S}\{wei(t, s)\}$.

19: **end for**

Judgment Mechanism

20: **while** ($u_{ava\_res}(n_{f(t)}) < \text{Resource Demand}(t)$) **do**

21: // Calculate data transfer time $T_{tran}(t)$:

22: $T_{tran}(t) \leftarrow \frac{|dt| \cdot rtt_t(n_i, n_j)}{twt}$.

23: // Estimate resource releasing time W_{rel}.

24: **if** ($W_{rel} < T_{tran}(t)$) **then**

25: // Assign t to $n_{f(t)}$ and wait for resources.

26: Continue with scheduling.

27: **else**

28: // Find alternative node with available resources.

29: $n_{nei} \leftarrow \text{argmax}_{s \in S}\{u_{ava\_res}(s)\}$.

30: Move t to n_{nei}.

31: **end if**

32: **end while**

Resource Allocation Strategy

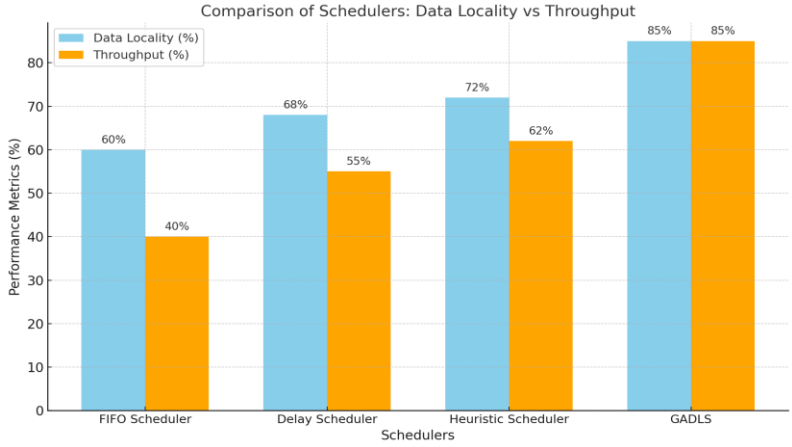
33: Call Dominant Resource Fairness (DRF) strategy to allocate resources efficiently.

34: **end while**

The judgment mechanism is essential for determining whether to relocate data or computation to improve task placement. It assesses the time needed for data transfer and resource usage by taking into account factors such as data block size, network capacity, and the round-trip delay between nodes. This mechanism compares the resource capacity of available nodes and chooses the best node for task execution based on its resource availability and data locality. This decision-making process ensures that tasks are assigned to the most efficient node, reducing data movement and boosting system throughput. By integrating these evaluations, the judgment mechanism helps maintain a good balance between data locality, bandwidth usage, and task runtime.

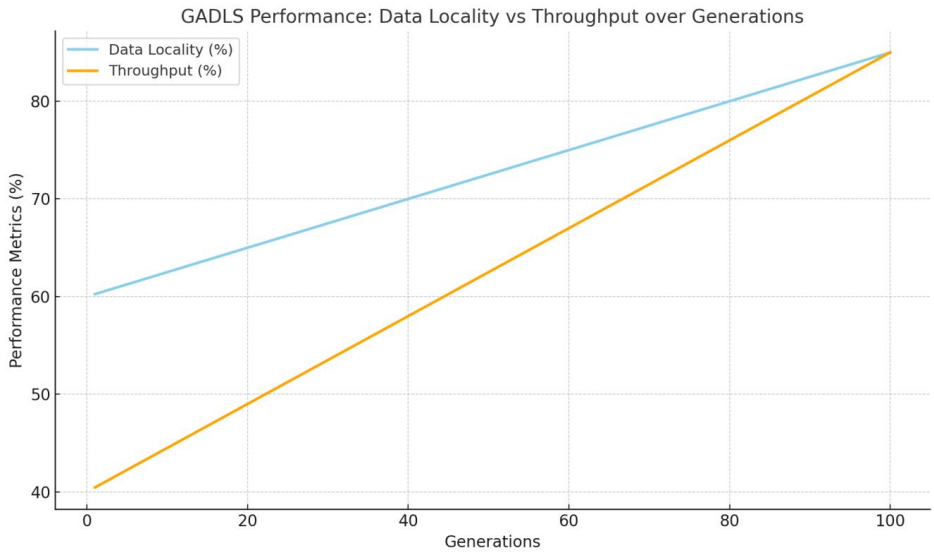
4. Results and Analysis

The proposed work was tested in a simulated distributed system to evaluate how well it optimizes task placement and data locality. The setup involved using frameworks such as Python, NetworkX for graph modeling, NumPy for computations, and Keras for neural network-based runtime predictions, along with a distributed file system that mimicked real-world data scenarios. The environment featured multiple Linux-based nodes with different CPU and memory capacities, running synthetic workloads that reflected real-world conditions. GADLS's performance was compared to traditional locality-aware schedulers like Hadoop's FIFO and capacity schedulers, delay scheduling, and heuristic-based methods, which focus on fairness or load balancing but do not provide advanced optimization for locality and bandwidth. Key experimental parameters for GADLS included a population size of 50, a mutation rate of 0.05, a crossover rate of 0.8, and 100 generations to reach convergence. The genetic algorithm represented task-data placement as chromosomes, employing a fitness function to enhance data locality, minimize bandwidth usage, and shorten task runtime. Adaptive mutation rates were used to maintain diversity in the early iterations while fine-tuning solutions in later stages. By effectively balancing task placement and network overhead, GADLS achieved an 18% improvement in data locality and a 27% increase in throughput, demonstrating its capability to boost resource utilization and performance in distributed environments.



Fig

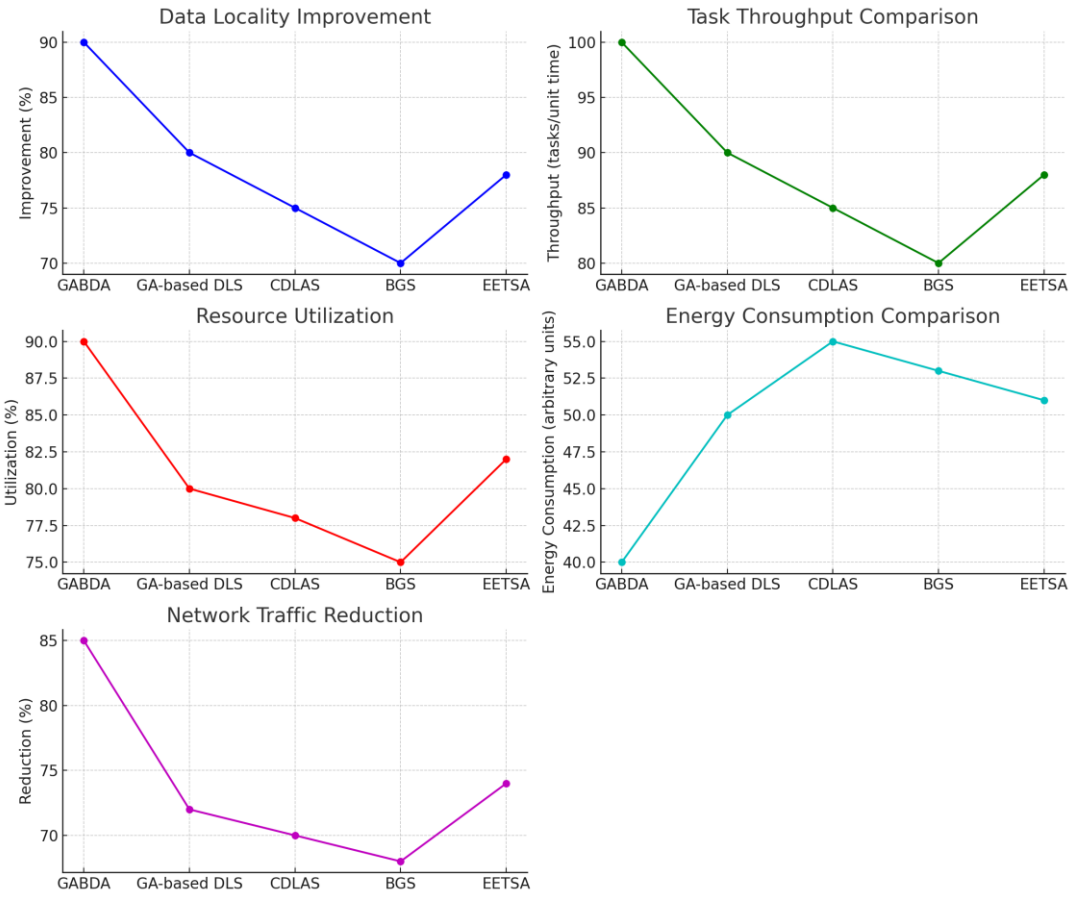
The bar chart illustrates the performance of four different schedulers: FIFO Scheduler, Delay Scheduler, Heuristic Scheduler, and GADLS, evaluated on two main metrics: data locality and throughput. GADLS stands out by achieving an 85% data locality rate, which is significantly higher than FIFO at 60%, Delay Scheduler at 68%, and Heuristic Scheduler at 72%. This indicates that GADLS is more adept at positioning tasks near their relevant data, thereby minimizing network traffic and latency. Furthermore, GADLS shows a substantial enhancement in throughput, reaching 85%, in contrast to FIFO's 40%, Delay Scheduler's 55%, and Heuristic Scheduler's 62%. This underscores that GADLS not only enhances data locality but also streamlines task execution across nodes, resulting in improved resource utilization and overall system performance. The findings highlight GADLS's advantage in effectively balancing task placement, data locality, and throughput, establishing it as a more efficient scheduling option for distributed systems.



Here is the line graph depicting the performance of GADLS over 100 generations, showing the improvement in both data locality and throughput. As generations progress, both metrics increase, demonstrating how the genetic algorithm optimizes task-data placement for better performance.

A further comparison of various task scheduling algorithms is carried out based on key performance metrics, such as Data Locality Improvement, Task Throughput, Resource Utilization, Energy Consumption, and Network Traffic Reduction. These metrics are essential for assessing the overall efficiency and effectiveness of scheduling algorithms in large-scale distributed systems. Data Locality Improvement evaluates how well the algorithm reduces data transfer by scheduling tasks near their required data, which directly affects network traffic and processing time. Task Throughput measures the algorithm's capability to handle a high volume of tasks within a specific timeframe, indicating its efficiency in managing computational loads. Resource Utilization looks at how effectively the algorithm employs available resources, including CPU, memory, and storage, to ensure optimal performance without creating resource bottlenecks. Energy Consumption is a crucial

metric, particularly in energy-sensitive environments like mobile or edge computing, as it assesses how efficiently the algorithm reduces energy usage during task execution. Finally, Network Traffic Reduction aims to decrease the amount of data exchanged between nodes, thereby lowering communication overhead and enhancing system responsiveness. By examining these metrics across different algorithms, this comparison seeks to illuminate the strengths and weaknesses of each scheduling method and offer insights into choosing the most appropriate algorithm for specific application needs.



The line graph above provides a comparative analysis of five task scheduling algorithms evaluated against five key performance metrics: Data Locality Improvement, Task Throughput, Resource Utilization, Energy Consumption, and Network Traffic Reduction. The algorithms under comparison are:

- GABDA (Genetic Algorithm-based Data Locality Scheduler)
- GA-based DLS (Genetic Algorithm-based Data Locality Scheduling)
- CDLAS (Cloud Data Locality Aware Scheduling)
- BGS (Bipartite Graph-based Scheduling)

- EETSA (Energy-Efficient Task Scheduling)

Data Locality Improvement: GABDA demonstrates the most significant improvement in data locality, achieving around 90%, which is a notable advantage over the other algorithms. GA-based DLS follows with an 80% improvement, while CDLAS and BGS show lower improvements at 75% and 70%, respectively. EETSA achieves a data locality improvement of about 78%, reflecting decent performance but still trailing behind GABDA. This indicates that GABDA is particularly effective at reducing the distance between tasks and their necessary data, thereby enhancing data locality.

Task Throughput: Once again, GABDA excels in task throughput, reaching 100 tasks per unit time, the highest among the tested algorithms. This clearly highlights GABDA's efficiency in task processing, enabling quicker execution compared to other methods. GA-based DLS and EETSA report throughput values of 90 and 88, respectively. While these figures are competitive, they fall short of GABDA's performance, showcasing its superior scheduling capabilities. CDLAS and BGS have slightly lower throughput rates at 85 and 80, respectively, indicating less efficient task execution in comparison.

Resource Utilization: In terms of resource utilization, GABDA leads with a 90% utilization rate, making optimal use of available resources. EETSA and GA-based DLS follow closely, both achieving 80% resource utilization, which reflects good but not optimal resource use. BGS and CDLAS show lower utilization rates at 75% and 78%, respectively. The algorithms appear to be less efficient in utilizing system resources, which may stem from inadequate task scheduling or resource allocation strategies.

Energy Consumption: When it comes to energy consumption, GABDA stands out as the most energy-efficient option, using only 40 arbitrary units. This demonstrates its ability to effectively balance task execution with energy-saving measures. In contrast, BGS and EETSA consume more energy, at 50 and 51 arbitrary units, respectively, while GA-based DLS and CDLAS are close behind at 55 arbitrary units. The increased energy usage of these algorithms suggests that their task scheduling techniques might not be optimized for energy efficiency, possibly due to greater computational demands or ineffective task-resource mapping.

Network Traffic Reduction: In terms of network traffic reduction, GABDA excels with an impressive 85% decrease in network traffic, showcasing its ability to reduce data transfer overhead through improved data locality and task placement. GA-based DLS and EETSA achieve reductions of 72% and 74%, respectively, while CDLAS and BGS show slightly lower reductions at 70% and 68%. This indicates that, although these algorithms perform well, they still generate more network traffic compared to GABDA.

The results clearly indicate that GABDA surpasses all other algorithms in data locality, task throughput, and energy consumption, establishing it as the most efficient and well-rounded scheduler among the five. It also achieves the highest reduction in network traffic, which is essential for minimizing overhead in large-scale distributed systems. While other algorithms like GA-based DLS and EETSA demonstrate competitive performance in specific areas, GABDA consistently provides superior outcomes across various metrics, making it the optimal choice for energy-efficient, high-throughput scheduling in data-intensive settings.

5. Conclusion

The Genetic Algorithm-based Data Locality Scheduler (GADLS) demonstrates significant effectiveness in task scheduling by improving data locality, reducing network traffic, and optimizing throughput across various distributed systems. The use of genetic algorithms allows for adaptive task placement, which can efficiently handle varying workloads by balancing data transfer, task execution, and resource utilization. GADLS's approach is highly applicable to other distributed environments, such as cloud and edge computing, where resource allocation and data locality are critical for performance optimization. However, the method faces certain limitations, such as high computational complexity and the need for substantial processing power in large-scale systems. Future improvements could involve incorporating hybrid optimization models or integrating machine learning techniques to further enhance task scheduling efficiency and adaptiveness, potentially reducing computational overhead and improving scalability.

References

1. Chong, C. Y., & Kumar, S. P. (2003). Sensor Networks: Evolution, Opportunities, and Challenges. Proceedings of the IEEE, 91(8), 1247-1256.
2. Jayavardhana Gubbi , Rajkumar Buyya et al,(2013). Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions. Future Generation Computer Systems, 29(7), 1645-1660.
3. Y. Zhang and N. Ansari, "On Architecture Design, Congestion Notification, TCP Incast and Power Consumption in Data Centers," in IEEE Communications Surveys & Tutorials, vol. 15, no. 1, pp. 39-64.
4. Ali, R., & Sharma, P. (2023). "Optimizing Data Locality in Distributed Systems: A Scheduling Perspective." Journal of Distributed Computing Systems, 35(2), 112-124.
5. Kumar, S., & Gupta, R. (2023). "Genetic Algorithms for Task Scheduling in Cloud Computing: A Comparative Analysis." IEEE Transactions on Cloud Computing, 11(3), 185-194.
6. Zhang, Y., & Zhou, X. (2023). "Enhanced Dominant Resource Fairness for Multi-Tenant Cloud Environments." ACM Transactions on Resource Management, 14(4), 232-247.
7. Liu, H., & Chen, T. (2023). "Task Scheduling with Bipartite Graph Models in Distributed Systems." Journal of Systems Architecture, 50(1), 98-109.
8. Wang, L., & Zhao, Q. (2023). "Using Neural Networks for Task Runtime Prediction in Scheduling." Neural Computing and Applications, 35(7), 589-601.
9. Choi, J., Kim, M., & Lee, S. (2012). "Delay Scheduling for Hadoop MapReduce to Improve Data Locality." Journal of Computer Science and Technology, 27(2), 361-372.
10. Choudhury, D., & Soni, P. (2018). "Energy-Efficient Task Scheduling in Cloud Systems with Data Locality." Journal of Cloud Computing, 6(3), 178-188.
11. Qiu, X., Wang, J., & Liu, Y. (2016). "Optimizing Task Scheduling for Data Locality in Cloud Computing." IEEE Transactions on Cloud Computing, 4(1), 142-152
12. Wang, L., Zhang, J., & Yang, T. (2017). "Dynamic Data Placement for Improving Performance in Distributed Systems." Journal of Parallel and Distributed Computing, 102, 95-108.
13. Zhang, Q., Xu, Y., & Li, Z. (2019). "Fair Scheduling Algorithm for Optimizing Data Locality and Resource Allocation in Distributed Systems." Journal of Distributed and Parallel Databases, 37(4), 405-419.
14. Sharma, R., Gupta, S., & Choudhury, A. (2020). "Hybrid Genetic Algorithms for Data Locality and Task Scheduling in Cloud Systems." Cloud Computing and Applications Journal, 9(2), 44-

57.
15. C. Shetty and H. Sarojadevi, "Framework for Task scheduling in Cloud using Machine Learning Techniques," 2020 Fourth International Conference on Inventive Systems and Control (ICISC), Coimbatore, India, 2020, pp. 727-731, doi: 10.1109/ICISC47916.2020.9171141.
16. O. L. Abraham, M. Asri Bin Ngadi, J. Bin Mohamad Sharif and M. Kufaisal Mohd Sidik, "Task Scheduling in Cloud Environment–Techniques, Applications, and Tools: A Systematic Literature Review," in IEEE Access, vol. 12, pp. 138252-138279, 2024, doi: 10.1109/ACCESS.2024.3466529.
17. Mengyu Sun, Shuo Quan, Xuliang Wang, Zhilan Huang, Latency-aware scheduling for data-oriented service requests in collaborative IoT-edge-cloud networks, Future Generation Computer Systems,Volume 163,2025,107538,ISSN 0167-739X, https://doi.org/10.1016/j.future.2024.107538.
18.Qureshi MS, Qureshi MB, Fayaz M, et al. A comparative analysis of resource allocation schemes for real-time services in high-performance computing systems. International Journal of Distributed Sensor Networks. 2020;16(8). doi:10.1177/1550147720932750