*Journal of Harbin Engineering University*
*ISSN: 1006-7043*
Vol 44 No. 9
September 2023

# Recommendation System for analyzing the effectiveness in Optimal Resource Allocation (ORA) thereby providing Green Computing.

**Dr. Ananthi Sheshasaayee [1] Mrs. K. Vasantha Meena [2]**
[1] Associate Professor & Head, PG & Research Department of Computer Science, Quaid-e-Millath Govt. College for Women (Autonomous), Anna Salai, Chennai 600 002. ananthi.sheshasaayee@qmgcw.edu.in, ananthi.research@gmail. com

[2] Assistant Professor, PG & Research Department of Computer Science,
Quaid-e-Millath Govt. College for Women (Autonomous), Anna Salai, Chennai 600 002.
www.qmgcw.edu.in, v.meenab4u@gmail.com

**ABSTRACT**
The digital world has revolutionized the entire human community. The usage of the various Information and Communication Technology (ICT) has led to an increase in the usage of electronic tools. These usage causes carbon emission at a large extent, which pollutes the environment. The carbon footprint left by these devices are growing day by day. Hence, there is a need to reduce this carbon emission by reducing the usage of such devices. In spite of carbon emission, the resources are left idle at many situations. The main aim of this paper is to propose a recommendation system that optimizes the resource utilization and allocates the resources thereby reducing the idle time of the devices. This helps in reducing the carbon emission and provide green computing.

## INTRODUCTION

Due to the recent increase in the use of digital devices, the environment is getting polluted. Hence, there has been a protest for going to green computing. Green computing, or sustainable computing, has been the talk of the day. It is the approach used to maximize energy efficiency and aims at minimizing the impact on the environment. This also has to do make changes in the way that computer chips, the systems, hardware and software are designed and used by people.

The green IT as it is also called, starts from the supply chain management in IT. It also looks into the ways through raw materials are acquired to build computers. Green computing also considers the way through which the systems are recycled. Systems developed with this approach, must work and deliver the expected output by using the least energy. More precisely, the energy consumed should be measured based on the performance per watt.

The first commercial use or motivation towards green computing was started by the year 2001 [1]. Wu Feng, a computer science professor at Virginia Tech, stepped on to green computing. As he was working at a National Laboratory in Los Alamos, he possessed a computer cluster for executing open science research. In the laboratory the professor maintained an external warehouse. This external warehouse had twice as many failures in summers when compared to winter. This situation made him to build a lower-power system that wouldn't generate as much heat. Wu Feng demonstrated the working of the system, termed it as "Green Destiny", at the Supercomputing conference in 2001. This program was covered by many famous channels like the BBC, CNN and the New York Times. This program sparked great talks and debates in the years to come together. It sparkled news about the potential reliability as well as efficiency of green computing.

As a solution to implementing green computing, the resources need to be used in an optimal manner. It has been observed that resources have been idle at times. To make an optimization, these unused idle resources can be pooled together and used. This paper aims to bring out a resource utilization that pools all the idle resources for usage thereby suggesting green computing.

## RELATED WORK

Resource management and allocation has been one of the ongoing research problems. Many have worked and proposed various solutions. A dynamic integration mechanism based on constraint programming is proposed in literature [2]. The existing algorithm, however, is mainly to study the internal allocation of resources on a single virtual machine. It is used to improve the utilization of physical resources and it was also possible to meet the users' computing requirements by dividing physical computing resources [3]. Reactive and Proactive approaches have been proposed for resource utilization. Table 1 [4] gives an overall view about the various solutions.

| Approach type | Control theoretic | Control structure | Prediction | Multiple resource allocation | Resource combination | Interference consideration | Solution |
|---|---|---|---|---|---|---|---|
| Reactive | - | - | ✓ | ✓ | - | - | VM migration |
| Reactive | - | - | - | - | ✓ | - | Threshold-based migration |
| Reactive | - | - | - | - | ✓ | - | VM assignment |
| Reactive | - | - | - | - | ✓ | - | Bidding language |
| Reactive | - | - | - | - | - | ✓ | VM instances allocation |
| Reactive | - | - | - | - | - | - | NP-hard optimization |
| Reactive | - | - | - | - | - | - | Linear programming |
| Proactive | - | - | ✓ | - | ✓ | - | Learning automation |
| Proactive | - | - | ✓ | - | - | - | Time series |
| Proactive | - | - | - | ✓ | - | ✓ | Reinforcement learning |
| Proactive | - | - | ✓ | - | - | - | Queuing theory |
| Proactive | - | - | - | - | ✓ | - | Access control |
| Proactive | ✓ | SISO | - | - | - | - | Feedback control, Optimal control, PID control |
| Proactive | ✓ | SISO | ✓ | - | - | - | Lyapunov technique |
| Proactive | ✓ | SISO | - | ✓ | - | - | Fuzzy control |
| Proactive | ✓ | SISO | ✓ | ✓ | - | - | Feedback control |
| Proactive | ✓ | MIMO | ✓ | - | - | - | Feedback control |
| Proactive | ✓ | MIMO | - | ✓ | - | - | LPV control, Stochastic model |
| Proactive | ✓ | MIMO | ✓ | ✓ | ✓ | - | Fuzzy control |
| Proactive | ✓ | MIMO | ✓ | ✓ | ✓ | ✓ | GPC control |

Table 1. Various solutions for resource utilization

Having a dynamic approach to solve the resource utilization has been a challenging one. Hence [5] – [13] have proposed models for handling the utilization in a dynamic manner using reactive approach. The proactive approaches use various algorithms like time series analysis [14] [15], queuing theory [16] [17] and access control [18][19]. All the proactive approaches [20 - 22] anticipates the future workloads by using all these algorithms. The solution proposed in this paper gives a dynamic solution for optimizing the resource utilization.

**PROPOSED MODEL**

The proposed ORA model solves the problem of resource utilization in a dynamic manner. The participants will be the users who are willing to share their idle resources. These users need to register themselves. All the details like the type of resources, their time of availability for the allocation and every other detail is gathered and maintained in a list. A learning mechanism is developed that will analyze this list to create a usage pattern. The usage pattern will have all the details of the idle resources, based on which the resources will be allocated.

The allocation of the resources will be done using a dynamic approach. For instance, a resource "X" has been allocated to a user, and during the execution if the resource "X" is being called by its owner, a new idle resource will be allocated to the user. This dynamic approach helps in continuing the work without any delay. The resources are allocated on the demand and on the usage pattern. Any queries are answered by the learning mechanism and any fault or unavailability of the resource is taken care.

The pooling of the resources and the allocation of the resources are performed using the "privacy preserving approach". The owner or the participant who is willing to share the resource will not know of who is using his resource or what is executed on the resource. The anonymity is maintained at both sides of the participant and the user. The architecture of the proposed ORA model is given in figure 1.

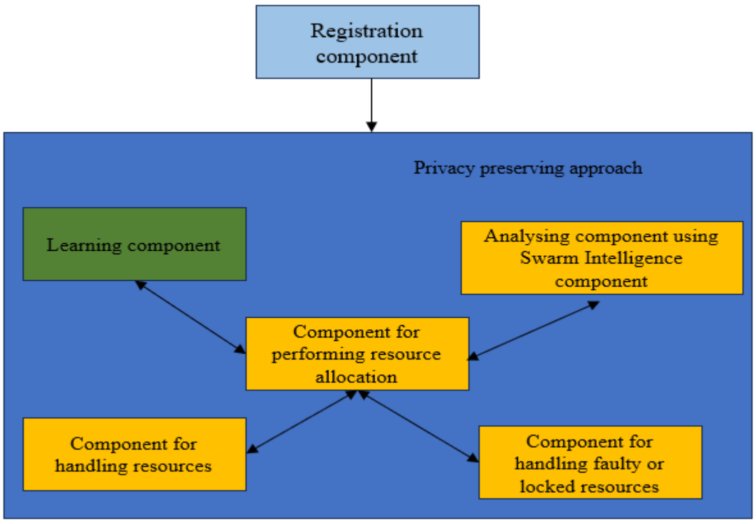


Figure 1. Architecture of the proposed ORA model

The above figure depicts an overall architecture of the proposed ORA model. Various components present in the proposed model are given. The overall working of the ORA model is deployed using the approach of "Privacy preserving" using which the anonymity is maintained. Each and every component performs a set of functions as discussed below:

(i) Registration component – The participants who are willing to share their resources register in the system. They enter the details of their resources and these details are maintained in a list

(ii) Learning component – A learning component is designed to learn the usage pattern and other details about the resources present in the list

(iii) Analyzing component using swarm intelligence – This component analyzes the list and tries to create a set of rules following which the resources can be allocated

(iv) Component for performing resource allocation – These are the exact components that perform the resource allocation using the demand and the usage pattern

(v) Component for handling faulty or locked resources – There may be cases when a resource which is under usage is needed by its owner or for any other reason for the resource to become faulty or get locked. This component handles all these problems and allocates a new resource so as to continue the execution. The dynamism of the ORA model is performed by this component

(vi) Component for handing queries – The participants may have certain queries regarding their resources or the user may have, this component handles all the queries.

The process flow of the proposed ORA model is given in figure 2 and the pseudocode is given in algorithm 1.

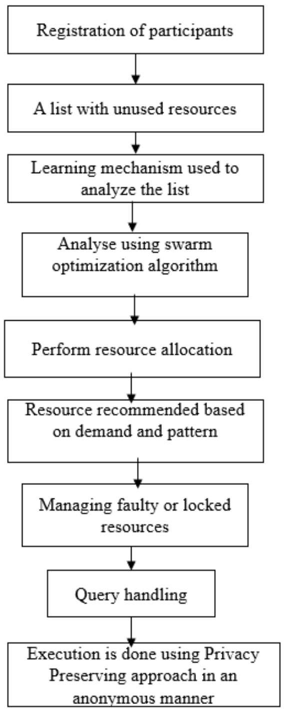


Figure 2. Process flow of the proposed ORA model

```
# Step 1: Registration of participants willing to give their unused resources
participants_list = []
def register_participants():
  while user_wants_to_register:
    participant = create_participant()
    participants_list.append(participant)
# Step 2: Gather and maintain full details of unused resources in a list
unused_resources_list = []
def gather_unused_resources_details():
  while user_wants_to_add_resource:
    resource_details = get_resource_details_from_user()
    unused_resources_list.append(resource_details)
# Step 3: Learning mechanism to study and analyze the working pattern of unused resources
learning_model = create_swarm_optimization_model()
def learning_mechanism():
  learning_model.train(unused_resources_list)
# Step 4: Analysis using swarm optimization algorithm
def swarm_optimization_analysis():
  best_resource_allocation = learning_model.optimize()
  display_results(best_resource_allocation)
# Step 5: Resource allocation based on demand and working pattern
def allocate_resources():
  task_demand = get_task_demand()
  best_resource = find_best_resource(task_demand, best_resource_allocation)
  allocate_resource_to_user(best_resource)

# Step 6: System recommends an unused resource for a task based on demand and pattern
def recommend_resource_for_task():
  task_demand = get_task_demand()
  best_resource = find_best_resource(task_demand, best_resource_allocation)
  display_recommendation(best_resource)
```

```
# Step 7: Manage faulty or locked resources
def manage_faulty_resource():
  if resource_faulty_or_locked:
    best_alternative_resource = find_next_possible_resource(task_demand,
best_resource_allocation)
    allocate_resource_to_user(best_alternative_resource)
# Step 8: Handle queries related to hardware requirements and unused resource allocation
def handle_queries():
  while user_has_queries:
    query = get_user_query()
    if query == "hardware_requirements":
      display_hardware_requirements()
    elif query == "user_unused_resources":
      display_user_unused_resources()
    else:
      display_invalid_query_message()
# Step 9: Execute everything with a Privacy Preserving approach
def privacy_preserving_execution():
  learning_mechanism()
  swarm_optimization_analysis()
  allocate_resources()
  recommend_resource_for_task()
  manage_faulty_resource()
  handle_queries()

# Step 10: Execute everything in an anonymous manner
def anonymous_execution():
  register_participants()
  gather_unused_resources_details()
  privacy_preserving_execution()
# Step 11: Main function to start the system
def main():

        anonymous_execution()
      # Call the main function to start the system
      main()
```

Algorithm 1. Pseudocode of the proposed ORA model

Above pseudocode gives the implementation part where the resource allocation is done. The main function calls the other related functions in a sequence as given in the figure 2. All the process and execution are carried out in an anonymous way. This helps in maintaining the integrity of the whole process.

**RESULT ANALYSIS**

The propose ORA model was implemented. The resource allocation was executed in an efficient manner. The performance of the proposed ORA is efficient when compared to other models. A comparison graph is given in figure 2. The overall performance is found to be improved as the learning component is used to make decisions etc. The ORA model is able to function efficiently as a dynamic approach is followed and hence there is no situation where the process stops.

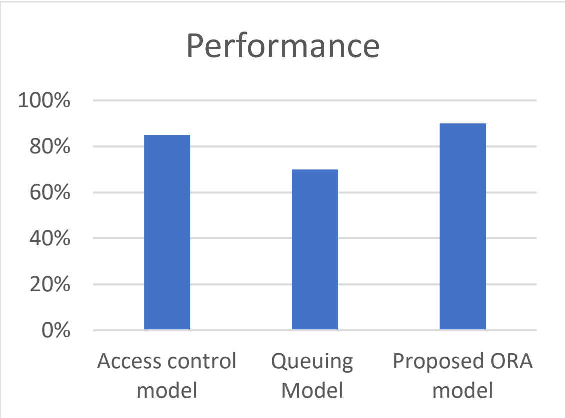


Figure 4. Comparison of performance of the ORA model with existing models

**CONCLUSION**

Thus, the resource utilization was carried out in an optimal manner. When the idle resources are pooled together and used, it decreases the carbon footprint set by the increase in usage of devices. As privacy preserving approach is used, the utilization of the resources are performed in a secure manner. In case of high demand, the proposed ORA model could do the over clocking whereby the resources are utilized to the maximum capacity.

**REFERENCES**

1. https://blogs.nvidia.com/blog/2022/10/12/what-is-green-computing/
2. M. E. Karim, M. M. S. Maswood, S. Das, and A. G. Alharbi, "BHyPreC: a novel Bi-LSTM based hybrid recurrent neural network model to predict the CPU workload of cloud virtual machine," IEEE Access, vol. 9, pp. 131476–131495, 2021.
3. D. B. Christie III, H. M. Nemec, A. M. Scott et al., "Early outcomes with utilization of tissue plasminogen activator in COVID-19-associated respiratory distress: a series of five cases," Journal of Trauma and Acute Care Surgery, vol. 89, no. 3, pp. 448–452, 2020.
4. S. Gong, B. Yin, Z. Zheng and K. -Y. Cai, "Adaptive Multivariable Control for Multiple Resource Allocation of Service-Based Systems in Cloud Computing," in IEEE Access, vol. 7, pp. 13817-13831, 2019, doi: 10.1109/ACCESS.2019.2894188.
5. Z. Xiao, W. Song and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment", *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1107-1117, Jun. 2013.
6. G. Zhang, X. Zhu, W. Bao, H. Yan and D. Tan, "Local storage-based consolidation with resource demand prediction and live migration in clouds", *IEEE Access*, vol. 6, pp. 26854-26865, 2018.
7. A. Beloglazov, J. Abawajy and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing", *Future Generat. Comput. Syst.*, vol. 28, no. 5, pp. 755-768, 2012.
8. A. G. Kumbhare, Y. Simmhan, M. Frincu and V. K. Prasanna, "Reactive resource provisioning heuristics for dynamic dataflows on cloud infrastructure", *IEEE Trans. Cloud Comput.*, vol. 23, no. 2, pp. 105-118, Apr./Jun. 2015.
9. L. Wang et al., "GreenDCN: A general framework for achieving energy efficiency in data center networks", *IEEE J. Sel. Areas Commun.*, vol. 32, no. 1, pp. 4-15, Jan. 2014.
10. H. Zhang, H. Jiang, B. Li, F. Liu, A. V. Vasilakos and J. Liu, "A framework for truthful online auctions in cloud computing with heterogeneous user demands", *IEEE Trans. Comput.*, vol. 65, no. 3, pp. 805-818, Mar. 2016.

11. L. Mashayekhy, M. M. Nejad, D. Grosu and A. V. Vasilakos, "An online mechanism for resource allocation and pricing in clouds", *IEEE Trans. Comput.*, vol. 65, no. 4, pp. 1172-1184, Apr. 2016.
12. M. R. Rahimi, N. Venkatasubramanian, S. Mehrotra and A. V. Vasilakos, "MAPCloud: Mobile applications on an elastic and scalable 2-tier cloud architecture", *Proc. IEEE UCC*, pp. 83-90, Nov. 2012.
13. S. S. Yau and H. G. An, "Adaptive resource allocation for service-based systems", *Proc. ACM Asia–Pacific Symp. Internetware*, pp. 3, Oct. 2009.
14. Z. Shen, S. Subbiah, X. Gu and J. Wilkes, "CloudScale: Elastic resource scaling for multi-tenant cloud systems", *Proc. ACM Symp. Cloud Comput.*, pp. 5, Oct. 2011.
15. R. N. Calheiros, E. Masoumi, R. Ranjan and R. Buyya, "Workload prediction using ARIMA model and its impact on cloud applications' QoS", *IEEE Trans. Cloud Comput.*, vol. 3, no. 4, pp. 449-458, Oct. 2015.
16. R. Tolosana-Calasanz, J. Diaz-Montes, O. F. Rana and M. Parashar, "Feedback-control & queueing theory-based resource management for streaming applications", *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 1061-1075, Apr. 2017.
17. M. Peng, S. Yan, K. Zhang and C. Wang, "Fog-computing-based radio access networks: Issues and challenges", *IEEE Netw.*, vol. 30, no. 4, pp. 46-53, Jul. 2016.
18. Z. Yan, X. Li, M. Wang and A. V. Vasilakos, "Flexible data access control based on trust and reputation in cloud computing", *IEEE Trans. Cloud Comput.*, vol. 5, no. 3, pp. 485-498, Jul./Sep. 2017.
19. M. Ali et al., "SeDaSC: Secure data sharing in clouds", *IEEE Syst. J.*, vol. 11, no. 2, pp. 395-404, Jun. 2017.
20. H. Morshedlou and M. R. Meybodi, "Decreasing impact of SLA violations: A proactive resource allocation approachfor cloud computing environments", *IEEE Trans. Cloud Comput.*, vol. 2, no. 2, pp. 156-167, Apr./Jun. 2014.
21. Z. Shen, S. Subbiah, X. Gu and J. Wilkes, "CloudScale: Elastic resource scaling for multi-tenant cloud systems", *Proc. ACM Symp. Cloud Comput.*, pp. 5, Oct. 2011.
22. R. N. Calheiros, E. Masoumi, R. Ranjan and R. Buyya, "Workload prediction using ARIMA model and its impact on cloud applications' QoS", *IEEE Trans. Cloud Comput.*, vol. 3, no. 4, pp. 449-458, Oct. 2015.