Wireless Personal Communications
https://doi.org/10.1007/s11277-021-08928-9

RDAD: An Efficient Distributed Multi-Layered Resource Handler in Jungle Computing

B. C. Manoj[1] · **D. Jeraldin Auxillia**[2]

Accepted: 8 August 2021

Abstract
The introduction of jungle computing raises many issues regarding its feasibility because of the distributed heterogeneous highly dynamic resources. Difference in resource representation in different platform poses the first hurdle. Discovering and allocating appropriate resources in jungle without much delay is a challenging task. The paper proposes a three layer distributed architecture that handles jobs separately and in parallel. Dedicated hardware devices that are connected to each other handle resource discovery and allocation. A uniform resource description framework helps reducing the latency for resource discovery. Additionally, the data migration delay is reduced by carefully considering the location of user, resource and dynamicity of the resource. The proposed method achieved better performance that the existing methods.

Keywords Jungle computing · Resource discovery · Resource allocation · RDAD · Data migration

1 Introduction

Distributed computing technology using distributed resources in cloud, cluster, grid and several standalone systems form jungle computing. Introduction of high performance CPUs, GPUs, ASIC and FPGAs to these infrastructures makes it computationally powerful. Jungle computing tries to utilize this computational efficiency of all these distributed environments. Jungle can perform computation irrespective of the resource location. Centralized control of resource modelling and allocation cannot work in jungle computing environment [1–3].

Due to high heterogeneity, dynamicity and hierarchy of jungle resources, a distributed architecture for resource description, discovery and allocation is required. The main issues in jungle computing is the lack of global organization of these services. The distributed

✉ B. C. Manoj
Manoj9444@gmail.com

D. Jeraldin Auxillia
Jeraldin.auxilia@gmail.com

[1] Department of CSE, St Xavier's Catholic College of Engineering, Nagercoil, Tamil Nadu, India

[2] Department of ECE, St Xavier's Catholic College of Engineering, Nagercoil, Tamil Nadu, India

architecture should able to do efficient resource description, resource provisioning, resource discovery, resource allocation and process migration. Additionally, each of these functionalities should be adaptive with the highly dynamic resources [3, 4].

The paper presents a distributed architecture for resource description, resource discovery, resource allocation and process migration in jungle computing. The architecture manages the jungles computing through a three level architecture. Resource description classifies the resources based on flexibility and performance. The resource discovery and allocation, which can take ample amount of time, has been moved to a separate layer that works on dedicated hardware. This will greatly improve the performance of jungle computing by allowing parallel execution of resource discovery and data computation.

2 Related Works, motivation and Contributions

Authors in [2] introduced a software platform Ibis, which can do computation on heterogeneous resources all around the world. The architecture takes care of the location of data and heterogeneity of resources to produce decent resource utilization. In [3], the authors formed a methodology, which selects the resources by minimizing the cost in terms of power and resource price. The system was tested with virtual machines, Raspberries and local clusters. Authors in [5] used jungle computing platform for identifying image source for forensic application. They then analysed the efficiency of different hardware in jungle.

Zarrin et al. [6] proposed Hybrid Adaptive Resource Discovery for Jungle Computing (HARD) for efficient resource discovery in jungle environment. The proposed method is proved to work on heterogeneous and dynamic environment. In [7], a case study on jungle computing for computational astrophysics, which uses multi-model/multi kernel approach.

Authors in [8] proposed ways to improve resource availability for mobile cloud computing. The issues of resource allocation, task scheduling and load balancing in cloud computing has been taken care in [9]. Resource scheduling for fog computing using every resource available by Bag-of-Tasks workload model has be presented in [10].

The major issue faced in jungle computing is the delay of finding proper resource that are highly dynamic and allocation of resource. In related works, authors have performed mechanism in software side to find proper resource and allocation of these resources. This will create extra computation and the advantage of jungle computing will be lost. Hence, in order to tackle the heterogeneous nature of jungle, functionalities should be moved to different layers. If each layer can be pipelined to work parallel, the delay incurred can be reduced. In addition, the data movement can be optimized based on location of user and the resource.

The contributions of the paper are:

- A distributed three-layer architecture for jungle computing for resource description, resource discovery, resource allocation and data migration
- Implements dedicated hardware- Resource Discovery and Allocation Device (RDAD) at the top layer for resource discovery and allocation
- Data transmission layer concerns about the transfer of data from source to destination
- A three layer resource description framework for classifying resources based on flexibility and efficiency
- An efficient data migration scheme which depends on location of user and the resource

3 Proposed System

The proposed architecture for efficient resource discovery, resource allocation and data transfer in jungle computing is viewed as a three layer processing as shown in Fig. 1. Layer 1 contains all the resources in jungle (CPUs, GPUs, FPGAs etc.). Layer 2 deals with data movement across jungle resources which is spread across the globe. The functionality of resource discovery and resource allocation happens at layer 3. The system performs all three layers functionalities in parallel. This will reduce the major issues in computation lagging that happens during merging of different environments.

3.1 Layer 1: Resource Modelling Layer

The current computing technologies are equipped with latest hardware resources like Graphics Processing Unit (GPU), Application Specific Integrated Circuit (ASIC) and Filed Programmable Gate Array (FPGA) apart from processors. Isolation of such heterogeneous environment results in underutilization of computing resources. Jungle computing integrates distributed computing environments like cloud, cluster, grid, supercomputers and many adhoc hardware. The concept aims to utilize distributed computing resources such as general-purpose resources (processors), special purpose resources (GPUs, ASIC) and accelerators (FPGA) effectively.

Transparency and efficiency of the jungle computing rely on how well resources are identified and allocated. This can be achieved by careful identification of resource type and resource location. In order to identify the exact resource in the jungle, the mapping scheme

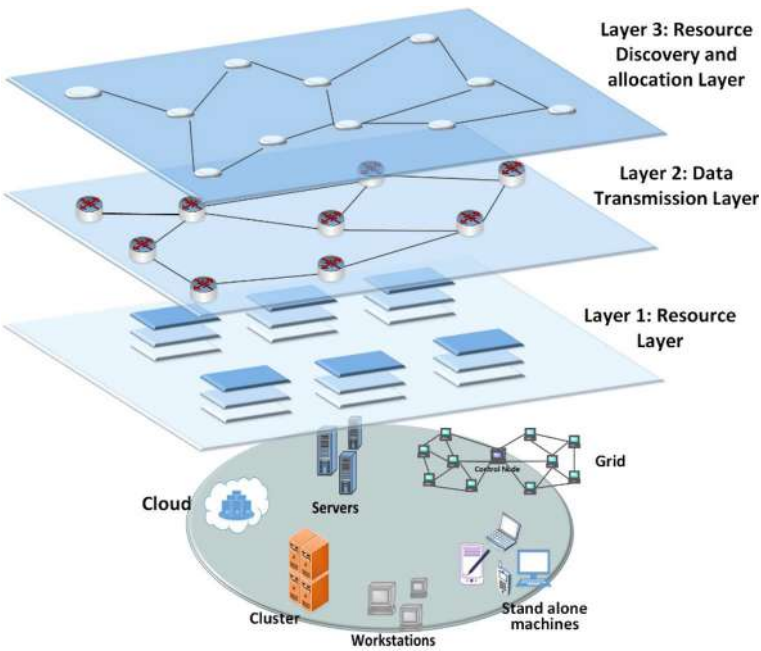


Fig. 1 Proposed architecture

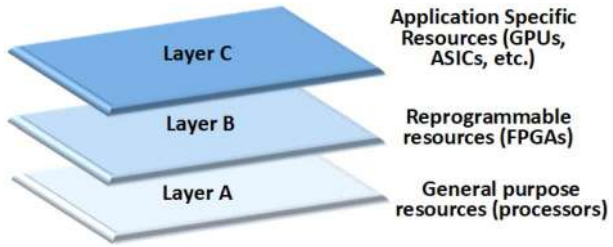


Fig. 2 Resource grouping in jungle computing

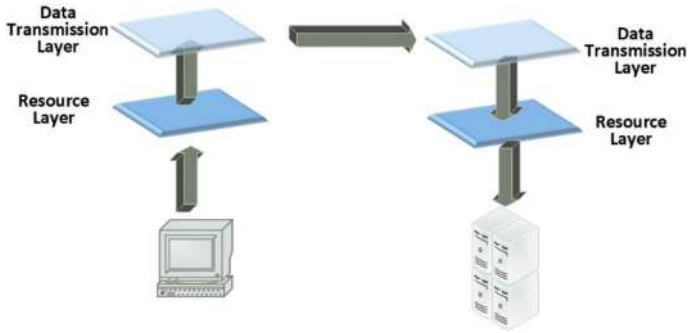


Fig. 3 Data Transmission layer

shown in Fig. 2 can be utilized. Layer A contains all the general purpose resource, layer B groups all the reprogrammable resources like FPGA and Layer C contains fast application specific resources.

Compared to Elcore [11], the proposed method could able to allocate the efficient resource from the layer. Elcore on the other hand, allocates the specific resource from the layers without considering the availability of ASIC or FPGA resources.

3.2 Layer 2: Data Transmission layer

Layer 2 deals with transfer of data from the user location to the resource location which will be identified by layer 3. After the resource is identified and allocated, user data will be passed from resource layer. Data passes from source to destination location as shown in Fig. 3. Once the resource location is confirmed, the data transmission layer works independently without disturbing the other functionalities.

3.3 Layer 3: Resource Discover and Allocation Layer

In order to perform resource discovery and allocation, dedicated hardware which is termed as Resource Discovery and Allocation Device (RDAD) is installed at each individual infrastructure (like cloud/cluster/grid etc.) in jungle. These RDADs are connected each other and runs resource discovery and resource allocation algorithm. After resource allocation has been done, user data is moved to the resource by data transmission layer.

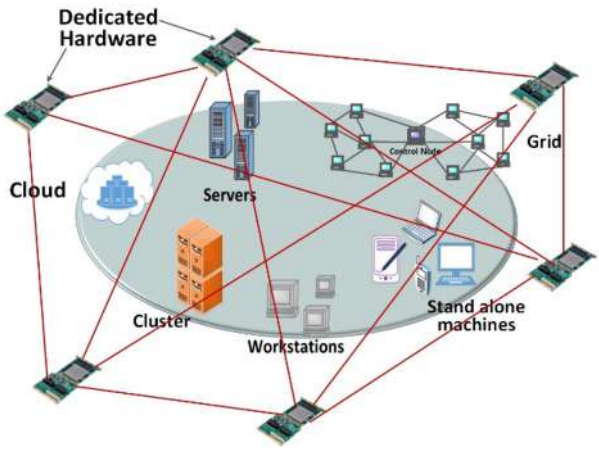
All the information regarding routing—routing table are stored in the RDAD. Whenever a new resource is being added to an infrastructure, the corresponding routing table is updated.

Fig. 4 Routing table inside the RDAD in layer 3

Routing Table

Resource	Route	Distance
Resource 1	Route 1	Distance 1
Resource 2	Route 2	Distance 2
.	.	.
.	.	.

Fig. 5 Interconnected hardware devices for resource discovery and allocation in layer 3



The format of routing table is shown in Fig. 4. It includes the resource type, route to the resource inside the infrastructure and distance of the resource from the network boundary of the infrastructure.

All the RDADs are interconnected with each other to form a peer-to-peer network as shown in Fig. 5. This allows performing distributed resource discovery and allocation to work smoothly.

As soon a request for computation arrives, it will be forwarded to the nearby RDAD. The device will execute resource discovery algorithm which in turn communicated with other devices. The resources and distance of resource from the request is being calculated. After the nearby appropriate resource is found, resource allocation algorithm allocates the resource.

4 Proposed Resource discovery and allocation mechanism

The proposed resource discovery and allocation process is implemented inside RDADs which are distributed across all platforms. The user request for a resource to the RDAD in its infrastructure. Layer 3 performs searching and allocation of resources. As soon as a

request is received, the concerned RDAD searches in its own platform. If the resource is not available, RDAD sends broadcast message to all the RDADs. Each RDAD searches for resource in the routing table and returns the location and timestamp if found. Timestamp will be required to authenticate the timing of reply and to avoid delayed messages. The source RDAD allocates nearest resource for the user. The steps are shown in Algorithm 1.

Algorithm 1: Resource Discovery and Allocation

Input: Resource Request, R_i
Output: Resource R allocated to request R_i

1. Receive resource request R_i
2. If found in Local pool
    a. Schedule the resource
3. Else do steps from 4 to 8
4. Broadcast_RDADs (R_i)
    a. Perform search_in_Remote_Routing_Table(R_i)
    b. If found, $reply_i$←get_reply($Location_i$/$Timestamp_i$)
5. For all i,
    a. Perform select_resource← shortest_distance($Location_i$, $Timestamp_i$)
6. Repy←Test(select_resource)
7. If reply = TRUE
8. Send data via data transmission layer

Figure 6 shows the pictorial representation of proposed resource discovery and allocation. Each infrastructure is equipped with an RDAD. The resource allocation is done after broadcasting the request to all other RDADs and receiving reply. After finding the nearest resource from the location information and timestamp, the status of the resource is again verified using a test message. If the resource is still available at the location, resource will be allocated and data transfer from user side will be initiated. This will take care of the dynamicity of the resource.

The proposed resource discovery and allocation method happens at layer 3 and can operate parallel with layer 1 and 2. Layer 3 operation is handled by RDADs and hence will not affect the proper working of lower layers. All RDAD is assigned an IP address and connected to each other. Hence the delay incurred in resource discovery and allocation will be negligible.

4.1 Proposed Data Migration Technique

In order to reduce the delay during data migration, the proposed method finds the shortest route from user to resource location before data transfer starts. The RDAD identifies the location of resource where computation has to be performed as described in Algorithm 1. After finding the location of nearby resource, the distance between user location and resource location is calculated. If the resource is situated near to the user, the shortest path will be set as path from user to resource location as shown in Algorithm 2.

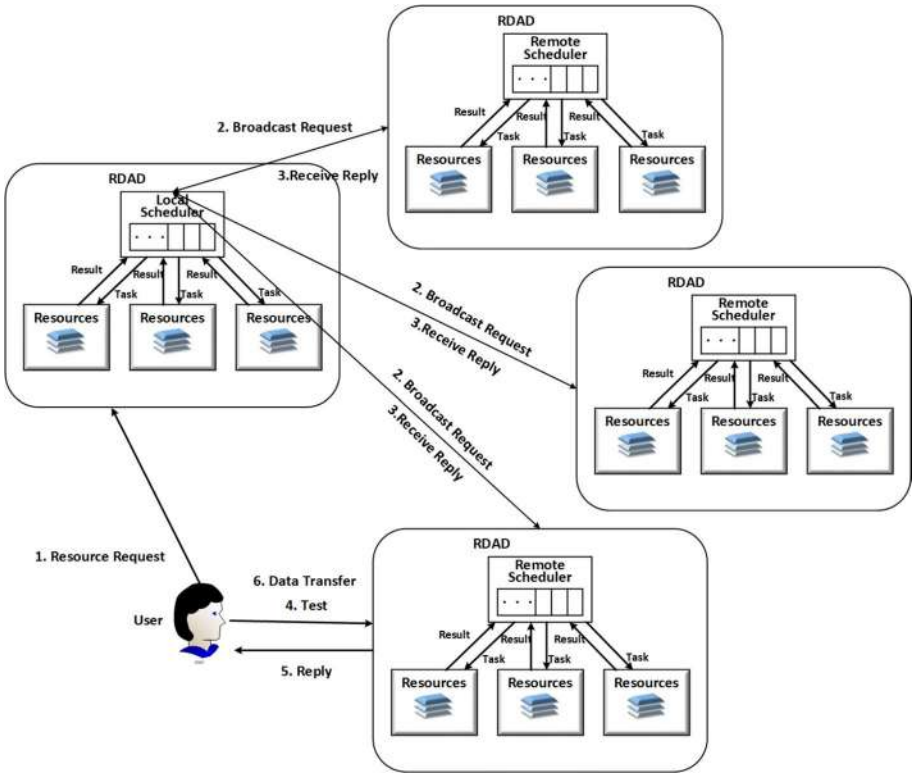


Fig. 6 Resource discover and allocation technique

This will reduce the delay in transmitting data and improve the performance of the system. Figure 7 shows the idea of eliminating longest path and selecting shortest path from the user to the resource.

Algorithm 2: Find shortest path

Input: Resource Request, RR_i
Output: shortest path, spath

1. Receive resource request, RR_i
2. Fetch $location_i$ from Algorithm 1
3. Get location of user, $location_u$
4. If distance ((current_location, $location_i$) >(($location_u$, $location_i$)
    a. Set spath← ($location_u$, $location_i$)
5. Else
    a. Set spath←(current_location, $location_i$)

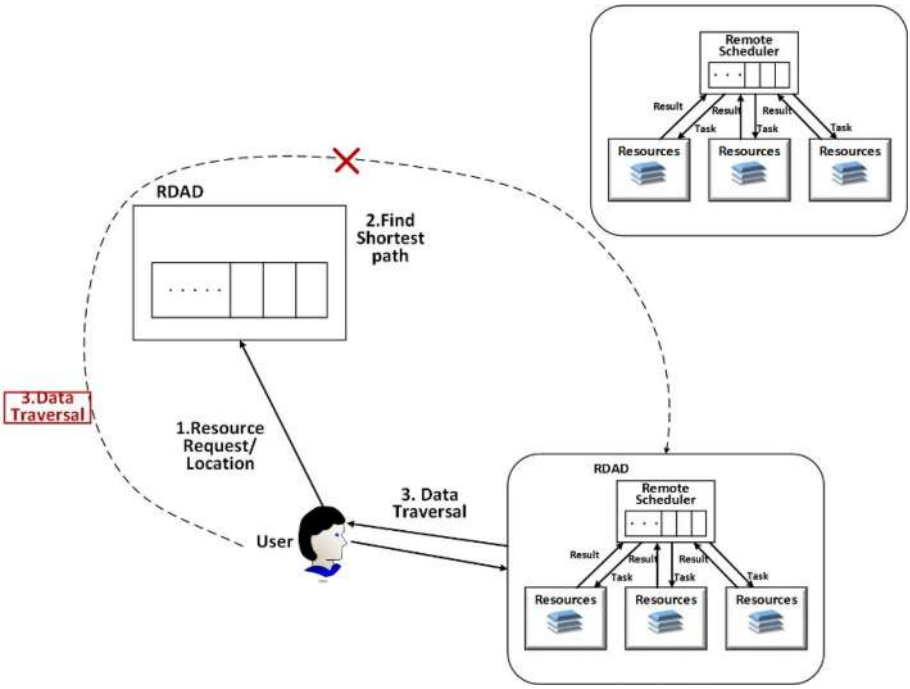


Fig. 7 Proposed data migration technique

Figure 8 shows the time multiplexing of proposed system in satisfying requests in jungle. Multiple requests can execute concurrently in RDADs depending on the number of instances in each platform.

Let the total number of RDADs in the jungle be N_{rdad}. Let T_d be the time for discovering a resource and T_a be the time for allocation, T_{dm} be the data migration time and N_r be the number of resource a particular RDAD can satisfy within time unit T.

Suppose $N_r = 30$, means a particular RDAD can satisfy 30 requests in one second. If there are 100 request arriving at a second, rate at which an RDAD satisfies request is $\Delta n_r = 0.3$.

Balance equation for the system can be written as:

Let ΔN_r be the small factor in satisfying the request

$$\Delta N_{rdad} = F\left(N_r + \Delta N_r\right) - F(N_r)$$

$\Delta N_{rdad} = n_r F\left(N_r\right)\Delta N_r - 0 \times F(N_r)\Delta N_r$, since the previous requests are not affecting the current requests

$$\Delta N_{rdad} = n_r F\left(N_r\right)\Delta N_r$$

$$\Delta N_{rdad} = 0.3F\left(N_r\right)\Delta N_r$$

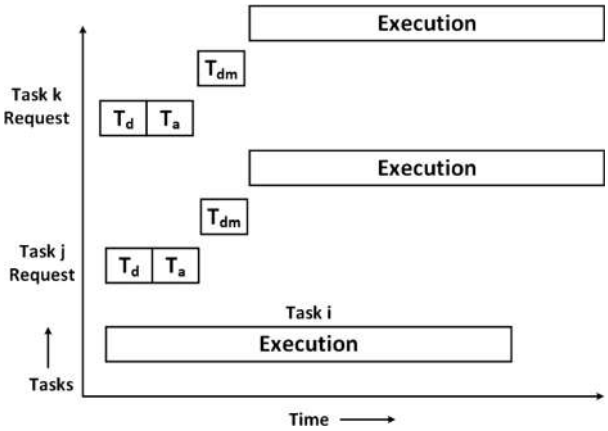


Fig. 8 Time multiplexing of satisfying requests in jungle

$$\frac{\Delta N_{rdad}}{\Delta N_r} = 0.3F(N_r) \tag{1}$$

$\frac{\Delta N_{rdad}}{\Delta N_r}$ is the small change in satisfying the incoming requests. Hence, to get the rate at which the RDADs satisfies the request, apply limit on Eq. (1).

$$\lim_{\Delta N_r \to 0} \frac{\Delta N_{rdad}}{\Delta N_r} = \lim_{\Delta N_r \to 0} 0.3F(N_r)$$

$$\therefore \frac{dN_{rdad}}{\mathrm{d}N_r} = 0.30F(N_r) \tag{2}$$

$\frac{dN_{rdad}}{\mathrm{d}N_r}$ is the rate at which the RDADs in jungle satisfies the requests

$$F(N_r) = \sum_{j=1}^{n} j \times n_{rd} \tag{3}$$

where n is the number of platforms in the jungle and n_{rd} is the number of RDADs allocated for each platform.

Substituting (3) in (2),

$$\frac{dN_{rdad}}{\mathrm{d}N_r} = \sum_{j=1}^{n} j \times n_{rd} \tag{4}$$

The rate of requests satisfied by RDADs depends on the number of platforms and RDADs inserted in the platforms.

Figure 9 shows the plot of Eq. (4) in the proposed system based on number on RDADs inserted at each platform (n_{rd}). The number of requests serviced substantially while inserting each RDAD. Hence, even though the jungle scales with the addition of more and more platforms, the proposed system scales fine with RDAD.

5 Implementation and Result Analysis

In order to verify the proposed system, we developed a simulation platform inside a server machine. Computing platforms that include cloud, cluster, grid and standalone systems are launched as virtual platforms inside the server. We have connected Xilinx Zynq-7000 SoC (xc7z020clg484-1) device as RDAD. In the simulation, all computing platforms are configured to share the same SoC. Resources in SoC are shared among different platforms to create five RDADs by Dynamic Partial Reconfiguration (DPR). For simulation, same SoC is used to implement all five RDADs for simplicity.

The RDAD module is developed in Xilinx High Level Synthesis (HLS). The module is imported to Xilinx Vivado to do partial reconfiguration and implementation [12–14]. Figure 10 show the design layout of five RDADs for cloud, cluster, grid and standalone systems. Each RDAD module is connected to BlockRAM (BRAM) to save routing table and resource details. BRAM is partitioned to provide multi-port. Multi-port BRAM increases parallelism by allowing parallel access of data inside it. Hence, the searching can be done faster. Figure 11 shows the chip layout of the design on xc7z020clg484-1 SoC using partial reconfiguration after placement and routing [15–17]. Using DPR, same SoC is shared and each RDAD can be loaded at run time depending on the platform usage.

The resource requirement of each RDAD is shown in Table 1. Since we are considering a test setup, a single SoC is placed at a location which is shared by all platforms.

The RDAD method increases parallelism by pipelining the resource discovery and allocation operation with lower layer functionalities. The proposed system is analysed with the resource allocation strategies of cloud, edge, grid and jungle environments.

Authors in [18] used a remote radio head (RRH) and cooperative communication and computation resource allocation (3C-RA) algorithm for low latency resource allocation for multi-layer cloud. Three 3C-RA model was tested for harsh resource allocation convergence condition and two models for medium convergence condition. The system is tested with multiple RRH for verifying the different methods. Figure 12 shows the comparison of the proposed methods with work presented in [18]. Using dedicated RDAD, the proposed system gave efficiently low latency in all conditions. The result shows the proposed RDAD gave additional support for cloud resource discovery compared to related works.

A balanced initialization, resource allocation and task allocation (BTR) allocation scheme is proposed in [19] for edge computing. In the work, particle swarm optimization

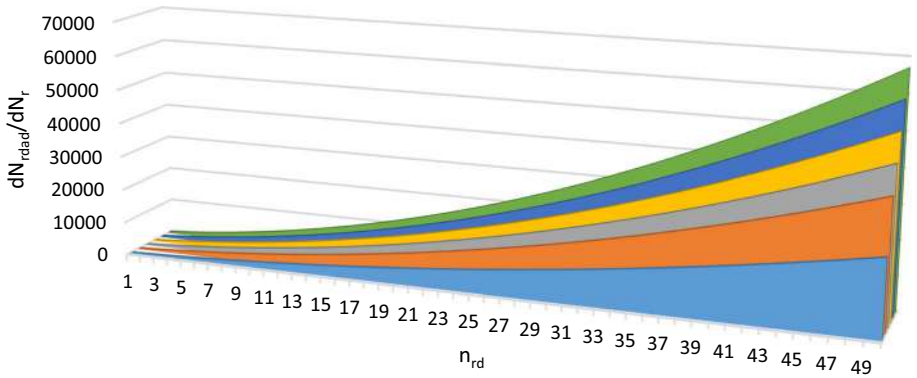


Fig. 9 Rate of satisfying the incoming requests based on number of RDADs in different platform

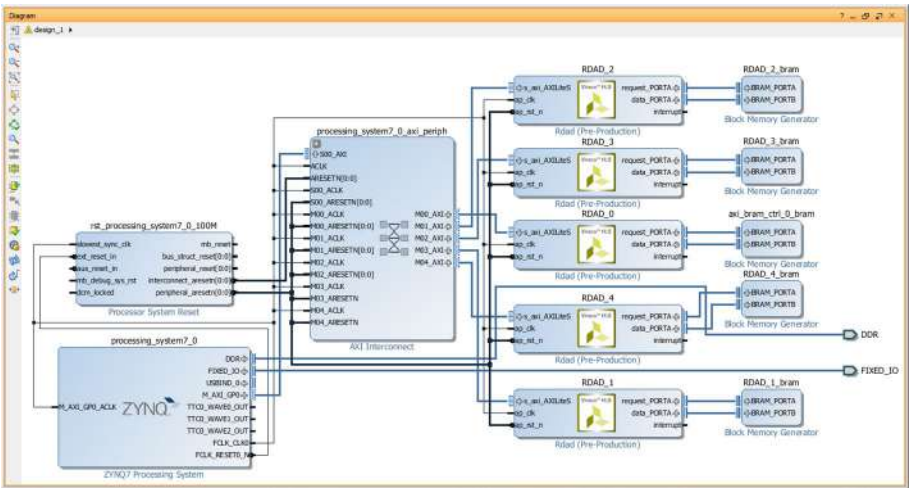


Fig. 10 Design layout of proposed RDAD for xc7z020clg484-1 SoC

technique is modified with pheromone strategy for efficient work allocation. The paper compared the proposed work with SAA-Simulated Annealing, LoAd Balancing (LAB) and Latency-awarE workloAd offloaDing (LEAD) [19–21]. The proposed system performance of using RDAD in edge environment is compared with all the methods and the results are shown in Fig. 13. Different Application programs selected was tested on the proposed system.

In grid environment, we have verified with the work done by Pujiyanta et. al [22]. The paper compared the proposed (FCFS-LRH) resource allocation method with FCFS-EDS [23], Aggressive backfilling, Backfilling [24] methods. Figure 14 shows the comparison of

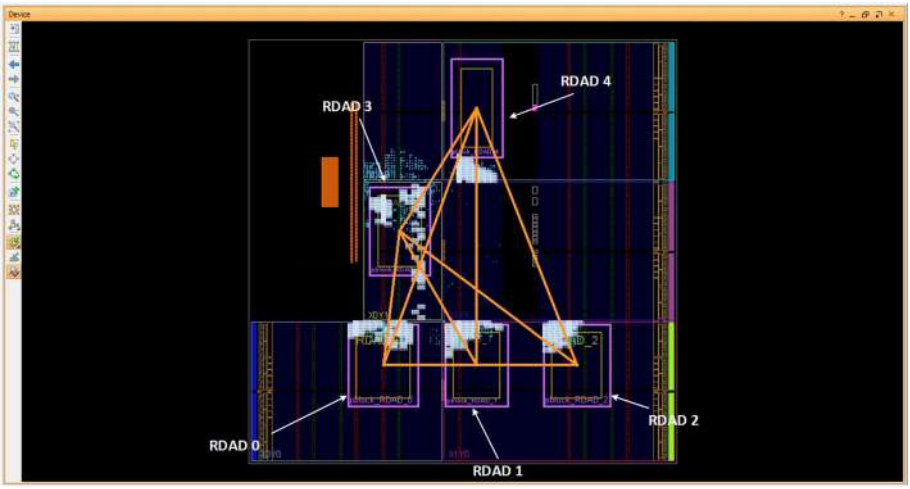


Fig. 11 Chip layout of RDAD0, RDAD1, RDAD2, RDAD3 and RDAD 4 on xc7z020clg484-1 SoC using Dynamic Partial Reconfiguration

Table 1 Resource requirement for a single RDAD

	BRAM_18k	FF	LUT	DSP48E
Routing_Table	10	33	43	0
Send_request	0	121	209	0
Search_resource	1	450	551	3
Receive_reply	0	78	82	0
Select_resource	0	110	243	0
Send_data	2	97	89	0
Update_resource	1	224	432	0
Total	14	1113	1649	3

proposed methods with [22]. It shows the waiting time for resource discovery and allocation for certain jobs submitted. [25, 26] gives more insight into resource discovery for more applications. Using the proposed RDAD scheme, the grid can do resource discovery and allocation more efficiently than related works.

Figure 15 shows the comparison of average latency values obtained for resource discovery and allocation among proposed work and related work [6]. Work proposed in [6] used software methodology for finding resources and hence lack parallelism and performance. Hence, with the increase in number of resources, the algorithm needed more time for discovering the resources. In the proposed work, each RDAD is being attached to dedicated BRAMs for storing resource information. The BRAM partitioning allowed parallel access of resource information. Hence, the increase in resource doesn't affect the system performance more.

The experimental setup was tested with proposed data migration technique. Figure 16 shows the average latency observed while applying shortest path technique in Algorithm 2 for data migration. The algorithm that does not used shorted path transferred data from user to local scheduler and from local scheduler to the remotely allocated resource. However, the proposed method efficiently transferred data from user end to the resource

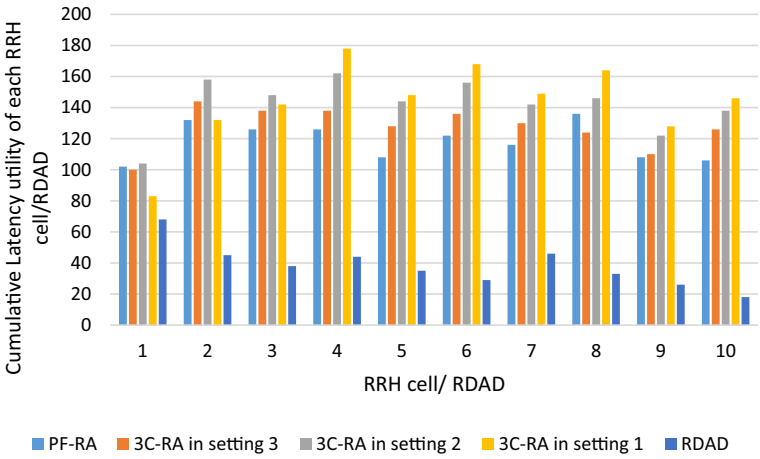


Fig. 12 Comparison of cumulative latency obtained in [18] and proposed work

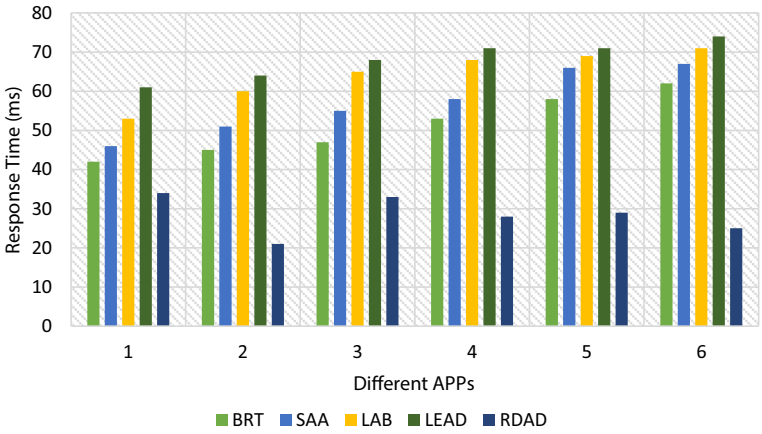


Fig. 13 Comparison of response time with different Application programs (APPs) of [18] and related work

location after confirming the shortest path. It is seen that the latency has reduced to a large amount while applying the proposed shortest path technique.

The results show that addition of extra dedicated hardware layer provided parallelism to resource discovery and allocation. This increases the response time and decreases the waiting time of resource allocation. The latency for resource discovery has reduced to a large extend due to achieved parallelism from dedicated hardware. Proposed data mitigation strategy reduces unwanted delays in traversal path which adds to improved response time.

6 Conclusion

The proposed method for resource description, resource discovery, resource allocation and data migration improved the overall performance of jungle computing. Unnecessary delays incurred heterogeneous environment is avoided by the introduction of proposed

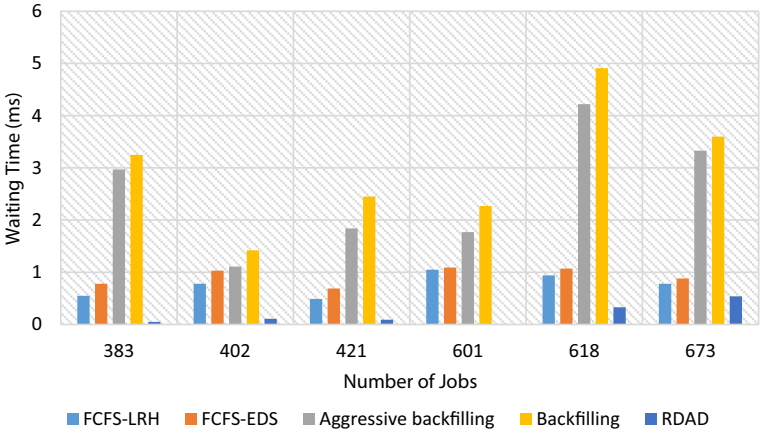


Fig. 14 Comparison of waiting time between [22] and proposed method

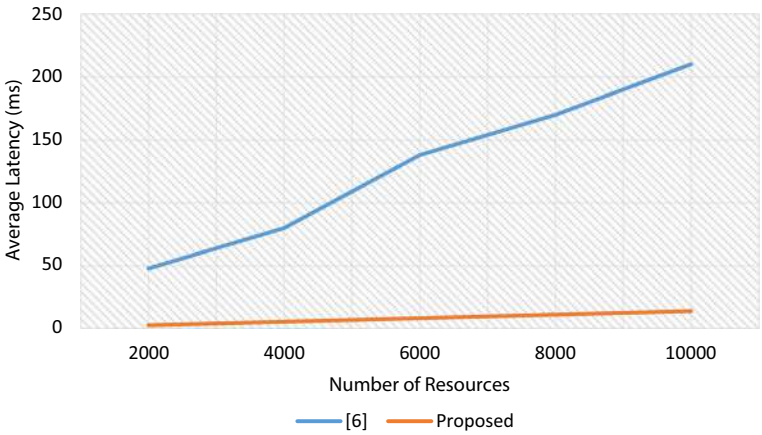


Fig. 15 Average latency observed for resource discovery and allocation

RDAD. The output analysis shows the introduction of dedicated hardware as a separate layer decreased the latency incurred compared to related works. The added hardware operates in a different layer and hence the resource discovery and allocation occurs in parallel. Additionally, proposed shortest path technique considered the proximity of user and resource location that make jungle computing more convenient. The proposed distributed framework increased the possibility of jungle computing usage in everyday life.

The future direction opens for the scope of developing a secure hardware platform for the entire resources in jungle. Having a secure hardware platform allows the safety of user program and data, which are executing on them. The secure jungle will be running on logically encrypted hardware platforms, which can only be operated with the help of an authorized key. Addition of secure hardware platform allows the acceptance of jungle computing for all highly secure applications as well.

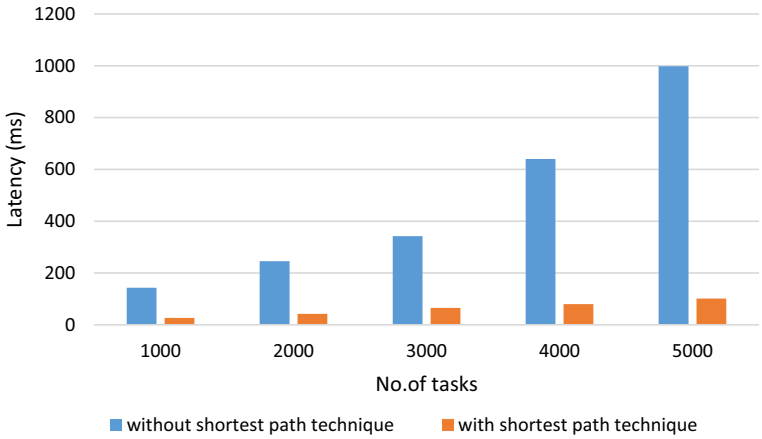


Fig. 16 Average Latency observed for data migration without and with proposed shortest path technique

References

1. Seinstra, Frank J., et al. (2011). "Jungle computing: Distributed supercomputing beyond clusters, grids and clouds." *Grids, Clouds and Virtualization*. Springer, London. 167–197.
2. Maassen, Jason, et al. (2011). "Towards jungle computing with Ibis/Constellation." *Proceedings of the 2011 workshop on Dynamic distributed data-intensive applications, programming abstractions and systems*.
3. Tychalas, Dimitrios and Helen Karatza. (2017). "High performance system based on Cloud and beyond: Jungle Computing." *journal of Computational Science* 22: 131–147.
4. Drost, Niels, et al. (2012). "High-performance distributed multi-model/multi-kernel simulations: A case-study in jungle computing." *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*. IEEE.
5. van Werkhoven, Ben, et al. (2018). "A Jungle Computing approach to common image source identification in large collections of images." *Digital Investigation* 27: 3–16.
6. Zarrin, Javad, Rui L. Aguiar and João Paulo Barraca. (2017). "HARD: Hybrid adaptive resource discovery for jungle computing." *Journal of Network and Computer Applications* 90: 42–73.
7. van Kessel, Timo, et al. (2014). "Toward a High-Performance Distributed CBIR System for Hyperspectral Remote Sensing Data: A Case Study in Jungle Computing." *High-Performance Computing on Complex Environments* : 401–428.
8. Thanikaivel, B., K. Venkatalakshmi and A. Kannan. (2021). "Optimized mobile cloud resource discovery architecture based on dynamic cognitive and intelligent technique." *Microprocessors and Microsystems* 81: 103716.
9. Asghari, Ali, Mohammad Karim Sohrabi and Farzin Yaghmaee. (2020). "Task scheduling, resource provisioning and load balancing on scientific workflows using parallel SARSA reinforcement learning agents and genetic algorithm." *The Journal of Supercomputing* : 1–29.
10. Tychalas, Dimitrios and Helen Karatza. (2020). "A scheduling algorithm for a fog computing system with bag-of-tasks jobs: Simulation and performance evaluation." *Simulation Modelling Practice and Theory* 98: 101982.
11. Zarrin, Javad, Rui L. Aguiar and Joao Paulo Barraca. (2017). "Decentralized resource discovery and management for future manycore systems.".
12. Vivado Design Suite Tutorial: High-Level Synthesis (UG871) – Xilinx (url: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2014_1/ug871-vivado-high-level-synthesis-tutorial.pdf).
13. Vivado Design Suite User Guide: Design Flows Overview – Xilinx (url: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2013_3/ug892-vivado-design-flows-overview.pdf).
14. Vivado Design Suite User Guide: Using Constraints (UG903) – Xilinx (url: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2013_1/ug903-vivado-using-constraints.pdf).
15. Vivado Design Suite User Guide Partial Reconfiguration, UG909 (v2016.1) (April 6, 2016) (https://www.xilinx.com/support/documentation/sw_manuals/xilinx2015_4/ug909-vivado-partial-reconfiguration.pdf).
16. Vivado Design Suite User Guide: Design Flows Overview --- Xilinx (https://www.xilinx.com/support/documentation/sw_manuals/xilinx2013_3/ug892-vivado-design-flows-overview.pdf).
17. Vivado Design Suite User Guide: Using Constraints (UG903) --- Xilinx (https://www.xilinx.com/support/documentation/sw_manuals/xilinx2013_1/ug903-vivado-using-constraints.pdf).
18. Mei, H., Wang, K., & Yang, K. (2017). Multi-layer cloud-RAN with cooperative resource allocations for low-latency computing and communication services. *IEEE Access, 5*, 19023–19032.
19. Niu, Xudong, et al. (2019). "Workload allocation mechanism for minimum service delay in edge computing-based power Internet of Things." *IEEE Access* 7: 83771–83784.
20. Pandit, Diptangshu, et al. (2014). "Resource allocation in cloud using simulated annealing." *2014 Applications and Innovations in Mobile Computing (AIMoC)*. IEEE.
21. Sun, X., & Ansari, N. (2017). Latency aware workload offloading in the cloudlet network. *IEEE Communications Letters, 21*(7), 1481–1484.
22. Pujiyanta, Ardi and Lukito Edi Nugroho. (2020). "Resource allocation model for grid computing environment." *International Journal of Advances in Intelligent Informatics* 6.2: 185–196.
23. R. Umar, A. Agarwal and C. R. Rao. (2012). "Advance Planning and Reservation in a Grid System," pp. 161– 173.
24. Moaddeli, H. R., Dastghaibyfard, G., & Moosavi, M. R. (2008). "Flexible Advance Reservation Impact on Backfilling Scheduling Strategies", in. *Seventh International Conference on Grid and Cooperative Computing, 2008*, 151–159.
25. Xia, Zhuoqun, et al. (2020). "Detection resource allocation scheme for two-layer cooperative IDSs in smart grids." *Journal of Parallel and Distributed Computing* .

26. Koole, G., & Righter, R. (2008). Resource allocation in grid computing. *Journal of Scheduling, 11*(3), 163–173.

B. C. Manoj obtained his Bachelor's degree in Computer Science and Engineering from Manonmaniam Sundaranar University, Tamil Nadu, India. Then he obtained his Master's degree in Computer Science and Engineering from Anna University, Chennai, Tamil Nadu, India. Currently he is a Full Time Ph.D scholar at St. Xavier's catholic college of Engineering, Anna University, Chennai, Tamil Nadu, India. His research interest includes high performance computing, cloud computing, grid computing, jungle computing and distributed computing.

D. Jeraldin Auxillia received the B.E degree in Instrumentation and Control Engineering from Government College of Technology, Coimbatore in 1988, the M.E Degree in Control and Instrumentation from the College of Engineering, Guindy, Anna University, Chennai in 2002, and the Ph.D from Anna University Chennai in 2012 in the area of controller design. She is currently working as Professor in department of Electrical and Electronics Engineering at St Xavier's catholic college of engineering, chunkankadai ,nagercoil ,tamilnadu, india. Her area of interest includes System identification and controller design, soft computing. She is a Life time member of ISTE.