Received: December 21, 2022. Revised: February 12, 2023. 486

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

High Reliability Through Cache Failure Minimization Technique for Workload Execution Under Multi-Core Edge-Cloud Platforms

Hrushikesh Joshi[1,2]* Uttam Patil[3] Kuldeep Sambrekar[4]

[1]*Jain College of Engineering, Department of Computer Science and Engineering, Visveswaraya Technological University, Belagavi, India*
[2]*Faculty of Engineering, Department of Information Technology, Society for Computer Technology and Research Pune Institute of Computer Technology, Pune, India*
[3]*Department of Computer Science and Engineering, Faculty of Engineering, Jain College of Engineering, Visveswaraya Technological University, Belagavi, India*
[4]*Department of Computer Science and Engineering, Professor, Karnataka Law Society Gogte Institute of Technology, VTU, Belagavi, India*
* Corresponding author's Email: hrushikeshjj@yahoo.com

Abstract: The Internet of Things (IoT) has become an infrastructure for various domain applications such as smart homes, smart cities, wearables, smart grids, etc. Due to these applications, the internet of things search engines has attained some attention from various researchers, industry, and users. The search throughput of IoT search engines is crucial because they must execute hundreds of spatial-time-keyword inquiries in a second. Moreover, IoT search engines use cloud resources to execute tasks. Some of the tasks might have a small workflow but some of the tasks might have a larger workflow and require more virtual machines. Instead of using the virtual machine the best option is to utilize a caching mechanism. In recent years, there are only a few methods that use the caching mechanism for the execution of the task. Furthermore, these methods are not reliable, and efficient and have not considered multi-core edge-cloud computing. To overcome this issue, this paper proposes a high-reliability method using the cache-failure mechanism for the multi-core edge-cloud computing architecture. The proposed High Reliability through cache-failure minimization (HRCFM) technique's main aim was to provide better tradeoffs during the execution of the task using the caching mechanism and reduce energy consumption. The results have been compared with the existing energy minimizing scheduling (EMS) and reliability (REL) technique in terms of execution time, power sum, power average, and energy consumption. The results have been compared and it shows that the proposed HRCFM technique reduces the time, energy consumption, power sum, and average by 83.3%, 90.92%, 38.44%, and 91.59% for EMS technique and 81.91%, 91.4%, 41.24% and 91.85% for REL technique respectively.

Keywords: Internet of things, Workflow, Cache failure, Reliability, Multi-core, Edge-cloud.

1. Introduction

Globally, organizations are adopting transformational change, although in most instances organizations should re-evaluate the capacity of their current technological infrastructures to satisfy the needs of information growth, edge development, IoT, and remote workers. As a result of the fact that information is created and consumed throughout multiple clouds, multi-core edge clouds, data centres, and other locations, there exists a substantial risk that data warehouses will form all across organizations [1]. This will restrict an organization's tendency to begin making an appropriate decision that are driven by data. Although the vast majority of information still is stored on-premises, it is anticipated that the number of different kinds of information which are gathered, analysed and maintained only at multi-core edge cloud outside of standard data centres or cloud infrastructure will rapidly increase in the not-too-distant future [2]. This is not an easy effort to manage

workstreams among all of those distant locations in addition to those that are located on-premises to assure every time connection, security, and compliance in the most cost-effective way possible [3]. It is necessary to have infrastructures, capacities, advising and consultancy services that allow businesses to handle, safeguard, and capitalize on all of their data, from the multi-core edge cloud platforms to the cloud [4]. Further, temporarily saving files or data enabling fast retrieving by users is nothing new; caching has served as the standard method for this for a long time [5]. As the need for digital resources increases around the world, from major cities to remote areas, edge computing has emerged as the leading method for bridging the gap between developers and online users [6]. Edge caching alleviates strain on large networks and boosts content distribution by relocating memory storage closest to end users [7]. This is achieved by establishing a structure in the memory space between central processing units, edge nodes, and local storage devices. In the edge computing environment, conventional data centers may have the most capacity, but their capabilities are rarely utilized by network end users [8]. Comparatively, edge servers can only store a little amount of data, but they contain files that are accessed often. Last but not least, local storage takes up the least space yet is where you'll find frequently accessed data and files [9]. As edge caching provides better results and there is a probability that there would be some failure during the retrieval of the information from the multi-core edge cloud. Hence, we propose a model which provides high reliability through cache failure minimization technique for workload execution under multi-core edge-cloud platforms. The proposed method comes with a multi-core edge-cloud platform that can execute workloads with a high level of reliability by employing a strategy to minimize cache failure. Further, to ensure that the presented multicore architecture and scheduling techniques are effective, we must first put them to the test using a variety of data-intensive and scientific workloads from the DAG.

In section 2, various studies have been conducted and the significance of the proposed work has been given. Further, in section 3, the proposed technique for scheduling scientific workload in a cloud computing environment using the caching technique has been given. In section 4, the results have been discussed with two existing methods. Finally, in section 5, the conclusion along with the future work has been given for the proposed model.

2. Literature survey

In [10], developed a resource optimizing approach for finding optimal system resources allotment to assure optimal availability and reliability at the minimum possible cost. This approach can utilize one of two resource optimization iterations, based on the traffic. Experiments outcome shows cooperative task scheduling is better than non-cooperative like Shortest-Job First-Scheduling and Earliest-Deadline First-Scheduling. However, the model failed reduce energy during task scheduling. In [11], they suggested an approach to minimize scheduling length, hence easing reliability constraints. To reduce energy consumption, they also suggested a processor-merging technique and a slack-time reclaiming method. The slack time method reduces task and processor speeds to save energy for execution of parallel workflows. However, the model to meet energy constraint result in increased execution time and doesn't provide information on how cache resource optimization can be done.

In [12], they develop S-Cache to provide edge cloud cache. They've established a novel cache-replacement technique to enhance hit rates. The developed cache-replacement technique outperforms least frequently (LFU) and greedy dual size frequency (GDSF). In each of these two databases, the aggregate cache hit-rates jump by 39% as well as the aggregate cache-replacement latency drops by 41% and 38%. multi-core edge cloud functionality, such as caching, computing, communication, and other processing, has been surveyed in [13]. The purpose of this review is to show how cache processing techniques can be essential in making this future a reality. In [14], the authors offer a video caching method built on multi edge-computing (MEC), wherein only certain maximum possible bit-rate video was cached and transcoded towards the desired lower bit-rate variant utilizing its processing capabilities just at MEC. They demonstrate that their approach improves the cache hit-rate, reduces the backhaul traffic burden, and speeds up video load time compared to the standard store and forward caching methodology. Nonetheless, how cache optimization meeting energy constraint is not addressed in their work. In [15], researchers analysed previous research from their perspective. In comparison to certain other literature reviews, theirs contains comprehensive research on the complete edge caching method, from policy development and caching location optimizations to content distribution. In [16], researchers worked on collaborative service-caching and task-offloading to improve edge-server and cloud-resource cooperation. In this approach,

they first framed this issue as mix-integer-nonlinear-programming, which really is NP-hard. Then, a three-stage heuristic for completing the task in polynomial time was proposed. In its early stages, this approach attempted to pre-offload as many tasks as possible to the cloud. Their method improved the efficiency of cloud-offloaded tasks by re-offloading part of them to edge-resources. Multiple mock experiments were used to evaluate their strategy. With respect to user happiness, resource effectiveness, and processing speed, their solution is up to 155%, 56.1%, and 155% better than both traditional and state-of-the-art alternatives respectively. However, these models have been tested using non-direct acyclic graph (DAG) application.

In [17], they have proposed an approach that explores Data, User, and power allocation (DUPA3) to serve as many consumers as necessary while maximizing the total data rate. First, they developed DUPA3 as a future game that had at least one Nash equilibrium and presented DUPA3. Further, in the results they explained that the dynamic cache reconfiguration improves energy consumption and efficiency in single-core computers. However, the model will exhibit poor performance for execution of DAG application. In [18], offers a static profiling-based method enabling vulnerability-aware energy-optimization in real-time multicore systems. Their method allows easy discovery of cache configurations and partition schemes for energy-optimization while meeting job constraints and vulnerability limits. Compared to state-of-the-art alternatives, their technique saves 19.2% more energy and reduces vulnerability by 49.3%. Nonetheless, work only for task with no interdependencies. In [19], researchers explored collaborative-edge data-caching (CEDC) issue to minimize network expenses including data caching, data migration, and QoS penalties. They developed a CEDC-O, an online method, to tackle the problem throughout all time periods. CEDC-O outperforms two main approaches on a real-world data set. Similarly, In [20], extended the work of [19] and proposed CEDC-IP for Integer-Programming CEDC. They also revealed the approximations ratio of CEDC-A, a method for quickly finding extensive CEDC solutions. Real-world and synthetic data are used to evaluate CEDC-IP and CEDC-A. The results show that the existing four popular approaches failed to match the proposed model. Nonetheless, the ILP formulation takes significantly higher computation overhead. In [21], they build a non-cooperative game among users, compute the corresponding Pure-Nash-Equilibrium (PNE), also known as the optimum data-offloading, and provide a decentralized low-complexity technique that responds to the PNE. Through modelling and testing, the efficiency of the proposed system as well as its fundamental principles are demonstrated. However, offloading of task doesn't consider energy constraint in scheduling reconfiguration. In [22], they examined cache-placement in Fog-RANs using adjustable transmission algorithms and user-content options. An approximate technique is offered to reach a solution around a constant-factor of either the best. In distributed system, a belief propagation-based distributed technique with iterative updates at each base station based on local data is provided. Simulations demonstrate that transmissions-aware caching can lower customers' average download delay by utilizing caching and cooperation improvements. However, the model failed to consider energy-makespan scheduling problem. Due to all these above issues found in the literature survey, we propose the significance of the research work as follows

1. One way that multi-core edge-cloud platforms can execute workloads with a high level of reliability is by employing a strategy to minimize cache failure.
2. To ensure that the presented multicore architecture and scheduling techniques are effective, we must first put them to the test using a variety of data-intensive and scientific workloads from the DAG.

3. Model

The purpose of this research is to develop a methodology for scheduling scientific workloads to reduce the number of Last-Level-Cache (LLC) failures. Because of the difficulty in allocating resources to several users, the cache-aware resource consumption algorithm was developed in this research. In the next stage, the computing-node is moved to a new location to cut down on LLC-failures in the distributed computing setting. The concept of allocation of resources consists of two stages. The first stage is for all virtual-computing-nodes (VCNs) to pool their cache memory and share it among the working nodes. This helps increase the system's throughput and storage space.

Further, the notation table of all the variables used in the equations has been denoted using Table 1.

When an LLC fails in the second stage of the computing environment, the scheduler will modify the virtual-computing-machines (VCMs) on the failing nodes. Algorithm 1 depicts the suggested technique operational flow. Specifically, the whole

Table 1. Notation Table

Variable	Definition
$P(t)$	Energy-dissipation
$e(t)$	Consumption of energy under the heterogenous cloud
M_a	Operational interval
B_k	Processor core
$v_k(t)$	Usage of processing core B_k
V_k	Resource consumption sets $V = [V_1, \dots\dots, V_i]^T$ for every frequency range $[R_{\uparrow,k,}, R_{\downarrow,k}]$ for every processor core B_k
$\{a_k(t) \mid 1 \le k \le i\}$	Cache-memory partitioning size
$\{\mathbb{f}_k(t) \mid 1 \le k \le i\}$	The operation frequency of the processing cores
$a_k(t)$	Size of the $L2$ *cache* partitions
$\mathbb{f}_k(t)$	Frequency of the processing element
$\mathbb{f}_{k\uparrow}(t)$	Peak frequency at a given B_k
A	The entire capacity of $L2$ *cache*
M_{kp}	Task operational time
$s_{kp}(t)$	Frequency-independent phase
D_{kp}, H_{kp} and X_{kp}	Quantifiable characteristics of jobs
X_{kp}	Size of the operational-set within that job
$h_k(t)$	Approximated processing-element resource utilization
q_{kp}	Job-rate within the operational time period M_{kp}
$\Delta h_k(t)$	Linear function with regard to $l_k(t)$
$\Delta a_k(t)$	Difference between $a_k(t)$ and $a_k(t-1)$
E	The estimated range for the device's pattern based on the computations performed
S_k, Y_k, and C_k	Processing element power consumption of VCN
$\mathcal{D}_g$	Caching benefits
$\mathcal{T}_g$	Size of the data being processed by the task
$\mathcal{U}_{seek}$	Time required to place the data in the cache partition
$\mathcal{Q}_g$	The ease with which the data may be retrieved from the cache
$\mathcal{D}_\downarrow$	The worst possible outcome of cache cost benefits
$\mathcal{D}_\uparrow$	Best possible outcome
$\mathcal{Q}_\ell$	Probability with which data blocks ℓ are accessed
$access_\ell$	The frequency with which data blocks ℓ are accessed
$\mathcal{Q}_{\ell_j}$	The probability of data blocks in the session x_j
σ	Partition size of data blocks for a certain job
$\mathcal{T}_\ell$	Standard data block size
$active_\ell$	The active mode of data blocks ℓ
$\mathcal{R}_\downarrow$	Least favourable outcome of the cost benefit of replacing the cache
$\mathcal{R}_\uparrow$	Most favourable outcome of the cost benefit of replacing the cache

VCMs at every processing-node are sorted as per LLC failures and afterward combined all together in compliance with respective failed LLCs of shared memory. Group A consists of all VCMs that experienced a catastrophic failure of their LLC, while Group B contains all VCMs that had the maximum number of L3 cache hits. Group A consists of VCMs that have a low rate of LLC failure, while Group B consists of VCMs that have a low rate of last-level caches. Work in the proposed scheduling model is carried out utilizing VCMs from either of the two categories. As a result, the scheduler can bring into existence not one but two types of VCMs, those with the fewest LLC failures and those with the most. In addition, the suggested model switches VCMs if there is a large disparity in the failure rates of the last few levels of caches. By gradually decreasing the frequency of cache failure only at the final level in a heterogeneous computing system, the suggested model accomplishes its scheduling in two stages.

Through the use of optimal cache optimization-based workload-scheduling, we provide a method for making the most of available resources in a cloud-computing environment that is heterogeneous (i.e., multi-core). By eliminating the dynamic boundary minimizing issue, the technique helps to lower the energy consumption of cloud-computing's computing nodes by decreasing their reliance on the cache. The following equation is used to calculate the energy-dissipation $P(t)$ in a heterogeneous computing system.

$$P(t) = e(t)M_a, \tag{1}$$

Where $e(t)$ represents the overall energy consumed under the heterogeneous-cloud conceptual model processing-element, which is mainly comprised of the operating frequency-level of the processor core B_k as well as the current $L2$ cache size. This is in contrast to $P(t)$, which presents the energy

Algorithm 1: Proposed technique for scheduling scientific workload in the cloud computing environment	
Step 1	***Start***
Step 2	***Compute*** and establish $N_{\mathbb{L}}$ (i.e., last level cache miss of each virtual computing machine (VCM).
Step 3	***Compute*** and establish $W_{\mathbb{L}}$ (i.e., last level cache miss of virtual computing machines in every processing node.
Phase 1	
Step 4	***For each*** processing node j from 1 to y **do** (i.e, establish the last level cache failure of each virtual computing machine in processing node j.
Step 5	$nx_j \leftarrow$ collects (j)
Step 6	$W_{\mathbb{L}} \leftarrow sort\ (nx_j)$ (i.e., arrange virtual computing node with last level cache failures).
Step 7	***Obtain*** $(W_{\mathbb{L}})$
Step 8	***End for***
Phase 2	Obtain processing node with maximal and minimal last level cache failures
Step 9	MaximalNode ← find MaximalNode($N_{\mathbb{L}}$)
Step 10	MinimalVCM ← findminimalVCM(MinimalNode) // establish (i.e., find) a virtual computing machine that is composed of maximal and minimal last level cache failures
Step 11	MaximalVCM ← findmaximalVCM(MaximalNode)
Step 12	MinimalVCM ← findminimalVCM(MinimalNode)
Step 13	**if** $T < maximalNode_{\mathbb{LLC}} - minimalNode_{\mathbb{LLC}}$ **then**
Step 14	interchange (maximalVCM,mimimalVCM)
Step 15	***Stop***.

consumed under heterogeneous cloud conceptual model processing-element for a specific (i.e., t^{th}) time duration. After that, the capacity of the $L2$ cache as well as the workload of the system will, for the most part, remain the same throughout the specified time of operation M_a. In this instance, M_a represents the operational interval to release multiple different information regarding every workload task during the process of t^{th} operational period.

The below equation can be used to determine the optimal cache partition-size and frequency to achieve a balance between minimizing energy usage and maximizing performance.

$$\min_{a_k(t)|1\le k\le i, \mathbb{f}_k(t)|1\le k\le i} \sum_{k=1}^{i}[V_k - v_k(t)]^2, \quad (2)$$

$$\min_{a_k(t)|1\le k\le i, \mathbb{f}_k(t)|1\le k\le i} P(t) \quad (3)$$

where $v_k(t)$ represents the usage of processing core B_k in the t^{th} operational period, V_k represents resource consumption sets $V = [V_1, \dots\dots, V_i]^T$ for every frequency range $[R_{\uparrow,k,}, R_{\downarrow,k}]$ for every processor core B_k, and to minimize variation in processor-core utilization $v_k(t)$ and utilization-sets (V_k), the cache-memory partitioning size is represented by $\{a_k(t)|1 \le k \le i\}$, and the operation frequency of the processing cores is represented by $\{\mathbb{f}_k(t)|1 \le k \le i\}$ at the tth operational period. There are 2 cache elements, $L1\ cache$ and $L2\ cache$ that make up the processor element of a cloud-computing system. In a multi-core sharing computing architecture, such caches are accessible by multicore processors at once. In this setup, every single processor is DVFS-ready. This will help save a considerable amount of the available energy resources. Cache-memory is divided up and used for many different purposes. Taking into account the amount of the processing-cores, B_k, $a_k(t)$ represents the size of the $L2\ cache$ partitions. The size of the peak frequency at a given B_k is denoted by $\mathbb{f}_{k_\uparrow}(t)$. The constraints given below in Eq. (4) and Eq. (5) must be met by Eq. (2) and Eq. (3).

$$R_{\downarrow,k} \le \mathbb{f}_k(t) \le R_{\uparrow,k}\ where, (1 \le k \le i) \quad (4)$$

$$\sum_{k=1}^{i} a_k(t) \le A \quad (5)$$

where A is the entire capacity of $L2\ cache$ that can be used in a cloud environment with varying hardware configurations. Eq. (2) shows the least amount of energy that will be wasted when carrying out a specific job inside a heterogeneous cloud computing architecture while using a specific amount of power generation $e(t)$ for the t^{th} operational period. The processor frequency is depicted by Eq. (3) and can be found inside the range of every processor core when the presented model is used. The difference in frequency-range is dependent on the type of processing-element that is being employed. The sum of all of the cache memory that has been partitioned is depicted by Eq. (5), and it is extremely close to being equal to the entire memory that is

accessible. Cache-aware resource utilization approach adjusts the caching partition-size and its core frequency for every processor core to minimize the difference between resource-utilization $v_k(t)$ and utilization sets V_k. Nevertheless, the operation time of a heterogeneous-computational system is impacted by the expense of optimizing frequency statically based on varied cache-partition sizes. Therefore, a dynamic optimizing approach is proposed to reduce processing time. Within t^{th} operational period, the model keeps the balance between $v_k(t)$, the central frequency $\mathbb{f}_k(t)$, and the optimization of feature $a_k(t)$. Specifically, the dynamic optimization model provides the best possible correlation between the parameters $b_{kp}(t)$, task operational time M_{kp}, optimizing feature $\mathbb{f}_k(t)$, and $a_k(t)$ for each core B_k in the t^{th} operational period. The following equation describes two possible strategies for optimizing the relation characteristic $b_{kp}(t)$, one of which is frequency independent.

$$b_{kp}(t) = s_{kp}(t) + i_{kp}.\left(\mathbb{f}_k(t)\right)^{-1}, \tag{6}$$

Since the computation time of input and output devices is not based on the frequencies of each core and $i_{kp}.\left(\mathbb{f}_k(t)\right)^{-1}$, the notation $s_{kp}(t)$ is used to describe the frequency-independent phase taking into account the relevant task operational time M_{kp}. Since the frequencies of the cores in operation affects the shape of the segment depicted by $i_{kp}.\left(\mathbb{f}_k(t)\right)^{-1}$, we call this the frequency dependent phase. $s_{kp}(t)$ is the amount of cache-memory set aside for each task's operational time M_{kp} when a given input and output device is excluded from the job execution. The optimum link between cache-failure and cache-memory size is mediated by the parameter $s_{kp}(t)$. The following equation provides an approximation of the optimal relationship between $s_{kp}(t)$, $a_{kp}(t)$, and assigned caches for the heterogeneous computing environment B_k.

$$s_{kp}(t) = \begin{cases} D_{kp}a_{kp}(t) + H_{kp} & 0 \le a_{kp}(t) \le X_{kp} \\ Constant & a_{kp}(t) \ge X_{kp} \end{cases} \tag{7}$$

where D_{kp}, H_{kp} and X_{kp} are quantifiable characteristics of jobs, and M_{kp} is an instance of a job operational session and X_{kp} is the size of the operational-set within that job. Eq. (7) demonstrates that the cache-memory capacity enhances and helps minimize the operation time period whenever the size of the operation set X_{kp} is larger than $a_{kp}(t)$,. Cache failure is also worse and cannot be fixed by adding more caching memory if the size of the operation set X_{kp} is smaller than $a_{kp}(t)$. The below equation establishes the relationship between the overall cache size $a_{kp}(t)$ provided to the processor core B_k in a heterogeneous computing system, the total independent frequency and operation session instance of each job in that system, and the total cache size $a_{kp}(t)$, of all systems in the system.

$$s_k(t) = \begin{cases} \sum_p D_{kp}{}' a_k(t) + \sum_p H_{kp} & 0 \le a_k(t) \le X_k \\ Constant & a_k(t) \ge X_k \end{cases} \tag{8}$$

where $D_{kp}{}' = \frac{D_{kp}a_{kp}(t)}{(a_k(t))}$ and $X_k = \sum_p X_{kp}$ Cumulating Eq. (7) for each task executed on the B_k. heterogeneous computational processor elements are depicted in Eq. (8). Therefore, the below equation describes how the suggested architecture helps to reduce interference in shared caches used by various processing elements.

$$h_k(t) = \sum_p i_{kp}\, q_{kp}\, .\left(\mathbb{f}_k(t)\right)^{-1} + \sum_p D_{kp}{}' q_{kp} a_k(t) + \sum_p H_{kp}\, q_{kp} \tag{9}$$

where $h_k(t)$ represents the approximated processing-element resource utilization and q_{kp} represents the job-rate within the operational time period M_{kp} for the heterogeneous computing system B_k. where $h_k(t)$ represents the approximated processing-element resource utilization and q_{kp} represents the job-rate. By utilizing Eq. (9), it is possible to demonstrate that $h_k(t)$ is proportionally inverse concerning the frequency of the processing element $\mathbb{f}_k(t)$. The following equation describes the estimated variance in resource use that occurs during $\Delta h_k(t)$ for the heterogeneous computing framework B_k.

$$\Delta h_k(t) = l_k(t) \sum_p i_{kp}\, q_{kp} + \Delta a_k(t) \sum_p D_{kp}{}' q_{kp} \tag{10}$$

where $\Delta h_k(t)$ is a linear function with regard to $l_k(t)$ which is defined as the difference between $\left(\frac{1}{\mathbb{f}_k(t)}\right)$ and $\left(\frac{1}{\mathbb{f}_k(t-1)}\right)$, and $\Delta a_k(t)$ is defined as the difference between $a_k(t)$ and $a_k(t-1)$. In Eq. (10), change the direct frequency utilization of the processing element $\mathbb{f}_k(t)$ to $l_k(t)$. The solution to

Eq. (10) demonstrates that the property $\Delta h_k(t)$ proportionate concerning i_{kp} and $D_{kp}{}'$ is true. As a result, the cost function of a heterogeneous computational environment can be minimized by utilizing a regulator for heterogeneous processing element called B_k and the equation that follows.

$$Z_k(t) = \sum_{c=1}^{E} \|v_k(t+c-1|t) - \beta\mathbb{f}_k(t+c-1|k)\|^2 + \|u_k(t|t) - u_k(t-1|t)\|^2 \quad (11)$$

where,

$$R_{\downarrow,k} \leq \mathbb{f}_k(t) \leq R_{\uparrow,k} \quad (12)$$

$$a_k(t) \leq a_{quota,k} \quad (13)$$

Where $\beta\mathbb{f}_k(t+1|t)$ is the pattern depicting how resource utilization influences/features $\beta\mathbb{f}_k(t+1|t)$ must transform its current utilization influence $v_k(t)$ to V_k, $u_k(t) = \begin{bmatrix} l_k(t) \\ \Delta a_k(t) \end{bmatrix}$. In case E of an operational session, E represents the estimated range for the device's pattern based on the computations performed. When using a heterogeneous computing architecture B_k, the cache size $a_k(t)$ is constrained by $a_{quota,k}$ to ensure Eq. (5). Consequently, the cache memory can be efficiently optimized by minimizing the least square problem using a dynamic model. The following equation describes an effective resource use model that can be used to optimize power consumption.

$$e_k(t) = S_k\mathbb{f}_k(t)^3 + Y_k a_k(t) + C_k \quad (14)$$

where,

$$R_{\downarrow,k} \leq \mathbb{f}_k(t) \leq R_{\uparrow,k} \quad (15)$$

$$a_k(t) \leq a_{quota,k} \quad (16)$$

Where S_k, Y_k , and C_k represent the processing element power consumption of VCN in a heterogeneous computing system. Various shared caches and processing elements contribute to the overall power consumption of a heterogeneous computational system. Power dissipation C_k and the dynamic power component $S_k\mathbb{f}_k(t)^3$ determine the overall power usage. Therefore, the proposed model can be used to reduce the amount of energy used by the cache memory. In addition, the caching cost is computed in this study so that the advantage of caching may be calculated. The larger the data, the greater the cost of caching it, therefore keep that in mind while estimating the benefit of caching your workload tasks' data. Cache benefits are often quantified by how much faster subsequent data accesses are. Therefore, caching is most beneficial when reaction times are reduced. As a result, the following equation is used to explain the benefits of data caching in terms of cost.

$$\mathcal{D}_{\mathcal{g}} = \begin{cases} 0 & \mathcal{Q}_g = 0 \\ \mathcal{Q}_g * \left(\mathcal{U}_{seek} + \frac{\mathcal{T}_g}{\mathcal{BW}_{cache}}\right) & \mathcal{Q}_g \neq 0 \end{cases} \quad (17)$$

Where $\mathcal{D}_{\mathcal{g}}$ depicts caching benefits, $\mathcal{T}_g$ represents the size of the data being processed by the task, $\mathcal{U}_{seek}$ represents the time required to place the data in the cache partition, and $\mathcal{Q}_g$ represents the ease with which the data may be retrieved from the cache. In this case, the equation below describes the maximum-minimum advantages of cache cost for fixing data comparability problems.

$$\mathcal{D}_{Ben} = \frac{(\mathcal{D}_{\mathcal{g}} - \mathcal{D}_{\downarrow})}{(\mathcal{D}_{\uparrow} - \mathcal{D}_{\downarrow})} \quad (18)$$

where $\mathcal{D}_{\downarrow}$ reflects the worst possible outcome of cache cost benefits and $\mathcal{D}_{\uparrow}$ the best possible outcome. There may also be a cost associated with clearing the cache and recaching the data. This work uses the number of unused (i.e. garbage) partitions in a data set as the metric for determining the cost of replacement. These data are less likely to be replaced if the respective data blocks are in active/hot mode. As a result, they have a lower replacement cost.

Taking into account window sampling x, the following equation can be used to define the access probability of each data block:

$$\mathcal{Q}_{\ell} = \frac{access_{\ell}}{access} \quad (19)$$

where $\mathcal{Q}_{\ell}$ describes the probability with which data blocks ℓ are accessed, $access_{\ell}$ represents the frequency with which data blocks ℓ are accessed, and access represents the total number of times data blocks ℓ are accessed in period x. Separate the session time into $x_1, x_2, \ldots, x_{\sigma}$ sub-windows. Following that, we determine the probability of accessing data blocks ℓ during the various parts of a session by doing the following.

$$\begin{cases} \mathcal{Q}_{\ell_1} = access_{\ell_1}/access_1 \\ \mathcal{Q}_{\ell_2} = access_{\ell_2}/access_2 \\ \quad\quad \ldots \\ \mathcal{Q}_{\ell_{\sigma}} = access_{\ell_{\sigma}}/access_{\sigma} \end{cases} \quad (20)$$

With $\mathcal{Q}_{\ell_j}$ representing the probability of data blocks in the session x_j, $access_{\ell_j}$ representing the frequency with which data block ℓ is accessed in the session x_j, and $access_j$ representing the overall frequency with which all data blocks are accessed in a session x_j. The following equation can be used to get the number of active mode data blocks, ℓ.

$$active_\ell = \frac{\mathcal{Q}_{\ell_2}}{\mathcal{Q}_{\ell_1}} * \frac{\mathcal{Q}_{\ell_3}}{\mathcal{Q}_{\ell_2}} * \frac{\mathcal{Q}_{\ell_4}}{\mathcal{Q}_{\ell_3}} * \frac{\mathcal{Q}_{\ell_5}}{\mathcal{Q}_{\ell_4}} \ldots * \frac{\mathcal{Q}_{\ell_\sigma}}{\mathcal{Q}_{\ell_{\sigma-1}}} = \frac{\mathcal{Q}_{\ell_\sigma}}{\mathcal{Q}_{\ell_1}} \quad (21)$$

As a result, the following equation can be used to describe the expense of data replacement

$$\mathcal{R}_g = \sum_{\ell=1}^{\sigma} \frac{\mathcal{Q}_\ell}{active_\ell * \mathcal{T}_\ell} \quad (22)$$

where σ represents the partition size of data blocks for a certain job, $\mathcal{T}_\ell$ represents the standard data block size, $active_\ell$ represents the active mode of data blocks ℓ, and $\mathcal{Q}_\ell$ represents the data block access probability for session x. In a manner analogous to the cache cost, we use Max-Min for the computation of the replacement cost as follows:

$$\mathcal{R}_{Ben} = \frac{(\mathcal{R} - \mathcal{R}_\downarrow)}{(\mathcal{R}_\uparrow - \mathcal{R}_\downarrow)} \quad (23)$$

where $\mathcal{R}_\downarrow$ illustrates the least favorable outcome of the cost benefit of replacing the cache, and $\mathcal{R}_\uparrow$ depicts the most favorable outcome of the cost-benefit of replacing the cache. Therefore, by utilizing the proposed model, we can bring about excellent tradeoffs between reducing the amount of energy that is lost and boosting the system performance of a heterogeneous cloud computing environment. This is experimentally demonstrated in the section that follows.

4. Results and discussions

In this section, the results have been compared with the existing EMS [11] method. The results have been compared in terms of total simulation time, power sum, power average, and energy consumption. The Inspiral workload is used for validating the performance of our proposed model and existing model. The Inspiral workload is memory and CPU intensive and is used for analyzing binary neutron stars and black holes. More details of workload can be obtained from [23].

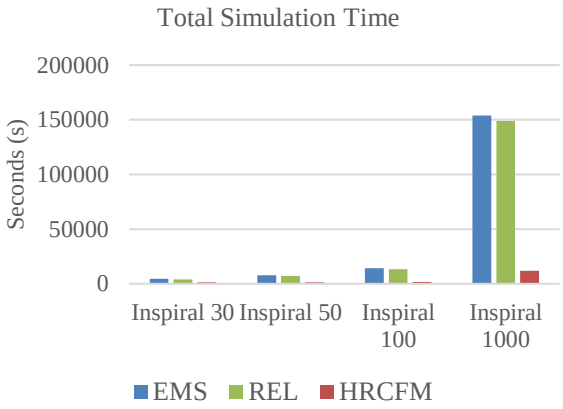

Figure. 1 Total simulation time

4.1 Total simulation time

In Fig. 1, the total simulation time required for the execution of the different number of tasks has been presented. The existing EMS [11] and REL [24] techniques have been compared with the proposed HRCFM technique. When compared with the existing EMS technique and HRCFM technique, the proposed HRCFM technique performs better by 69.93%, 82.125%, 89.23%, and 92.22% for Insipral 30, 50, 100 and 1000 respectively for total simulation time. Further, when compared with the REL technique, the HRCFM technique performs better by 66.57%, 80.44%, 88.67%, and 91.97% for Insipral 30, 50, 100, and 1000 respectively for total simulation time. The proposed HRCFM technique consumes less time for the execution of the task when compared with the existing EMS and REL techniques by 83.3% and 81.91% respectively.

4.2 Power sum

In Fig. 2, the power sum required for the execution of the different number of tasks has been presented. The existing EMS [11] and REL [24] techniques have been compared with the proposed HRCFM technique. When compared with the existing EMS technique and HRCFM technique, the proposed HRCFM technique performs better by 82.37%, 89.51%, 93.68%, and 98.13% for Insipral 30, 50, 100, and 1000 respectively for power sum. Further, when compared with the REL technique, the HRCFM technique performs better by 83.44%, 89.88%, 93.74%, and 98.29% for Insipral 30, 50, 100, and 1000 respectively for power sum. The proposed HRCFM technique consumes less power sum for the execution of the task when compared with the existing EMS and REL techniques by 90.92% and 91.34% respectively.

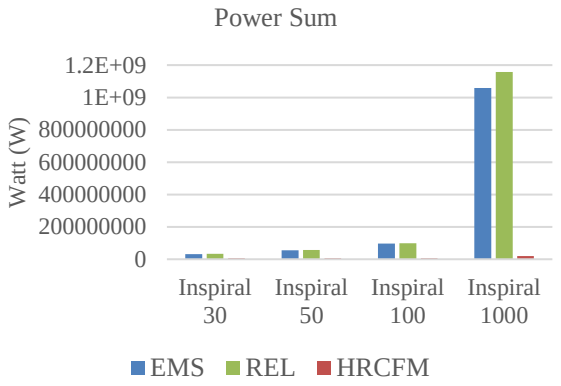

Figure. 2 Power sum

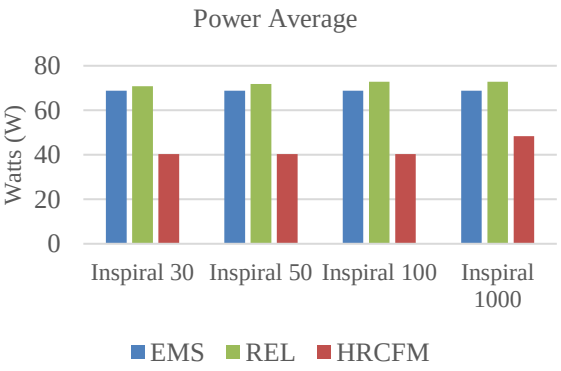

Figure. 3 Power average

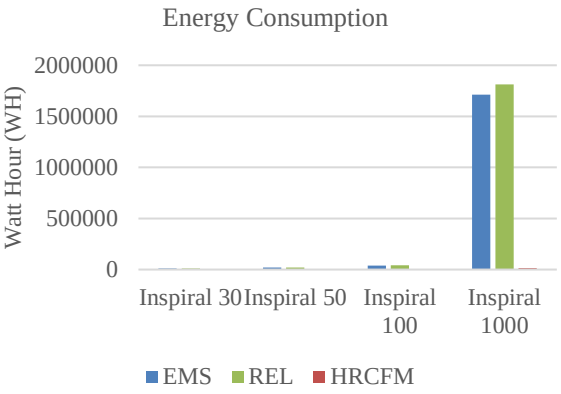

Figure. 4 Energy consumption

4.3 Power average

In Fig. 3, the power average required for the execution of the different number of tasks has been presented. The existing EMS [11] and REL [24] techniques have been compared with the proposed HRCFM technique. When compared with the existing EMS technique and HRCFM technique, the proposed HRCFM technique performs better by 41.357%, 41.356%, 41.355%, and 29.72% for Insipral 30, 50, 100, and 1000 respectively for power average. Further, when compared with the REL technique, the HRCFM technique performs better by 43.01%, 43.80%, 44.57%, and 33.58% for Insipral 30, 50, 100, and 1000 respectively for power average. The proposed HRCFM technique consumes less power on average for the execution of the task when compared with the existing EMS and REL techniques by 38.44% and 41.24% respectively.

Table 2. Comparison Table

	EMS [11]	**REL [24]**	**HRCFM [Proposed]**
Reliability	No	Yes	Yes
Multi-Cloud	No	Yes	Yes
Energy Optimization	Yes	No	Yes
Workload Type	Small-Large	Small-Large	Small-Large
Cache Utilization	No	No	Yes

4.4 Energy consumption

In Fig. 4, the energy consumed for the execution of the different number of tasks has been presented. The existing EMS [11] and REL [24] techniques have been compared with the proposed HRCFM technique. When compared with the existing EMS technique and HRCFM technique, the proposed HRCFM technique performs better by 83.271%, 90.40%, 93.53.355%, and 99.19% for Insipral 30, 50, 100, and 1000 respectively for energy consumption. Further, when compared with the REL technique, the HRCFM technique performs better by 83.61%, 90.88%, 93.69%, and 99.23% for Insipral 30, 50, 100, and 1000 respectively for energy consumption. The proposed HRCFM technique consumes less energy for the execution of the task when compared with the existing EMS and REL techniques by 91.59% and 91.85% respectively.

4.5 Discussions

From the results, it can be seen that the proposed HRCFM model attains better performance in terms of energy consumption, power average, power sum, and total cost. Further, the comparison table has been given in Table 2. The EMS [11], model has not considered reliability and a multi-cloud environment for the execution of the tasks. Further, the REL [24] model has not considered the energy optimization problem. Also, the EMS and REL models have not used any caching technique in their models. The proposed model provides reliability in a multi-cloud environment considering energy optimization techniques for the execution of the tasks. Also, the

proposed HRCFM technique has used caching technique. Due to this, the proposed HRCFM model is better than the existing models.

5. Conclusion

In this research work, first, a survey has been done on various edge cloud computing research works. Further, we have surveyed various cache failure minimization techniques which will help us to provide significance for the proposed research work. A system has been proposed to provide high reliability through cache failure during the execution of various tasks or workloads. After this, the model has been experimented with using the Inspiral workflow. The results have been compared with the existing Energy Minimizing Scheduling (EMS) and Reliability (REL) technique in terms of execution time, power sum, power average, and energy consumption. The results have been compared and it shows that the proposed HRCFM technique reduces the time, energy consumption, power sum, and average by 83.3%, 90.92%, 38.44%, and 91.59% for EMS technique and 81.91%, 91.4%, 41.24% and 91.85% for REL technique respectively. Moreover, our model provides better performance when the number of tasks are more. For future work, we would consider other scientific workflows such as Montage, SIPHT, and Epigenomics for the evaluation of the proposed HRCFM technique.

Conflicts of interest

The authors declare no conflict of interest.

Author contributions

Hrushikesh Joshi and Dr. Uttam Patil conceived the presented idea. Hrushikesh Joshi developed the theory and performed the computations. Dr. Uttam Patil and Dr. Kuldeep P. Sambrekar verified the analytical methods. Dr. Uttam Patil and Dr. Kuldeep P. Sambrekar encouraged Hrushikesh Joshi to investigate new methods for improving the models and supervised the findings of this work. All authors discussed the results and contributed to the final manuscript.

References

[1] B. Wang, C. Wang, W. Huang, Y. Song, and X. Qin, "A Survey and Taxonomy on Task Offloading for Edge-Cloud Computing", *IEEE Access*, Vol.8, pp.186080-186101, 2020.

[2] I. Ayo, T. Williams, and J. Yahaya, "Cloud Management and Monitoring-A Systematic Mapping Study", *Indonesian Journal of Electrical Engineering and Computer Science*, Vol. 21, No. 3, pp.1648, 2021.

[3] M. Allakonda and K. Sagar, "A Survey on data security challenges in multi cloud environment", In: *Proc. of 2021 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, pp. 1-5, 2021.

[4] A. Chandrashekhar and S. Suryakanth, "A Combined Computing Framework for Load Balancing in Multi-Tenant Cloud Eco-System", *International Journal of Electrical and Computer Engineering*, Vol. 12, No. 5, pp. 5630, 2022.

[5] D. Abramson, J. Carroll, C. Jin, M. Mallon, Z. Iperen, H. Nguyen, A. McRae, and L. Ming, "A Cache-Based Data Movement Infrastructure for on-Demand Scientific Cloud Computing", *Supercomputing Frontiers*, pp.38–56, 2019.

[6] W. Stallings, "An Overview of Cloud Computing", *Cloud Computing Security*, pp. 13–30, 2020.

[7] Y. Tang, D. Rajendiran, and M. Moh, "Cache Management for Cloud RAN and Multi-Access Edge Computing with Dynamic Input", In: *Proc. of 2019 International Conference on High Performance Computing & Simulation (HPCS)*, pp. 716-723, 2019.

[8] K. Cao, Y. Liu, G. Meng, and Q. Sun "An Overview on Edge Computing Research", *IEEE Access*, Vol. 8, pp. 85714–85728, 2020.

[9] W. Suadi, S. Djanali, and R. Anggoro, "Ghost-Cache CRFP Algorithm Simulator Based on Trace and File System", *Indian Journal of Computer Science and Engineering*, Vol. 12, No. 4, pp. 827–832, 2021.

[10] A. Amer, I. Talkhan, R. Ahmed, and T. Ismail, "An Optimized Collaborative Scheduling Algorithm for Prioritized Tasks with Shared Resources in Mobile-Edge and Cloud Computing Systems", *Mobile Network Applications*, Vol. 27, pp. 1444-1460, 2022.

[11] B. Hu, Z. Cao, and M. Zhou, "Energy-Minimized Scheduling of Real-Time Parallel Workflows on Heterogeneous Distributed Computing Systems", *IEEE Transactions on Services Computing*, Vol. 15, No. 5, pp. 2766-2779, 2022.

[12] C. Huang and S. Shen, "Enabling Service Cache in Edge Clouds", *ACM Transactions on Internet of Things*, Vol. 2, No.3, pp. 1-24, 2021.

[13] S. Barbarossa, S. Sardellitti, E. Ceci, and M. Merluzzi, "The Edge Cloud: A Holistic View of Communication, Computation, and Caching",

Cooperative and Graph Signal Processing, pp. 419-444, 2018.

[14] S. Kumar, D. Vineeth, and A. Franklin "Edge Assisted Dash Video Caching Mechanism for Multi-Access Edge Computing", In: *Proc. of 2018 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pp. 1-6, 2018.

[15] H. Wu, Y. Fan, Y. Wang, H. Ma, and L. Xing, "A Comprehensive Review on Edge Caching from the Perspective of Total Process: Placement, Policy and Delivery", *Sensors*, Vol. 21, No. 15, p. 5033, 2021.

[16] X. Chen, T. Gao, H. Gao, B. Liu, M. Chen, and B. Wang, "A Multi-Stage Heuristic Method for Service Caching and Task Offloading to Improve the Cooperation between Edge and Cloud Computing", *PeerJ Computer Science*, Vol. 8, p. 1012, 2022.

[17] X. Xia, F. Chen, Q. He, G. Cui, J. Grundy, M. Abdelrazek, X. Xu, and H. Jin, "Data, User and Power Allocations for Caching in Multi-Access Edge Computing", *IEEE Transactions on Parallel and Distributed System*s, Vol. 33, No. 5, pp. 1144-1155, 2022.

[18] Y. Huang and P. Mishra, "Vulnerability-Aware Energy Optimization Using Reconfigurable Caches in Multicore Systems", In: *Proc. of 2017 IEEE International Conference on Computer Design (ICCD)*, pp. 241-248, 2017.

[19] X. Xia, F. Chen, Q. He, G. Cui, J. Grundy, M. Abdelrazek, X. Xu, and H. Jin, "Online Collaborative Data Caching in Edge Computing", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 32, No. 2, pp. 281–294, 2021.

[20] X. Xia, F. Chen, J. Grundy, M. Abdelrazek, H. Jin, and Q. He, "Constrained App Data Caching over Edge Server Graphs in Edge Computing Environment", *IEEE Transactions on Services Computing*, Vol. 15, No. 5, pp. 2635-2647, 2021.

[21] P. Apostolopoulos, E. Tsiropoulou, and S. Papavassiliou, "Risk-Aware Data Offloading in Multi-Server Multi-Access Edge Computing Environment", *IEEE/ACM Transactions on Networking*, Vol. 28, No. 3, pp. 1405-1418, 2020.

[22] J. Liu, B. Bai, J. Zhang, and K. Letaief, "Cache Placement in Fog-RANs: From Centralized to Distributed Algorithms", *IEEE Transactions on Wireless Communications*, Vol. 16, No. 11, pp. 7039-7051, 2017.

[23] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M. Su, and K. Vahi, "Characterization of scientific workflows", In: *2008 Third Workshop on Workflows in Support of Large-Scale Science*, Austin, TX, pp. 1-10, 2008.

[24] X. Tang, "Reliability-Aware Cost-Efficient Scientific Workflows Scheduling Strategy on Multi-Cloud Systems", *IEEE Transactions on Cloud Computing*, Vol. 10, No. 4, pp. 2909-2919, 2022.