Quantifying the Effectiveness of Cloud and Edge Offloading: An Optimization Study on Energy Efficiency of Mobile Real-Time Systems

Gahyeon Kwon
Dept.of Computer Engineering
Ewha University
Seoul, Republic of Korea
sonaby2@ewhain.net

Hyokyung Bahn
Dept.of Computer Engineering
Ewha University
Seoul, Republic of Korea
bahn@ewha.ac.kr

***Abstract* — The rapid growth of IoT and AI technologies has driven the increased use of mobile embedded systems with real-time constraints. These battery-powered systems should minimize energy consumption while meeting task deadlines. Optimizing the trade-off between energy consumption and performance is crucial, particularly as workload demands fluctuate. Previous research has focused on optimizing system resources, such as processors and memory, by configuring low-power modes based on workload intensity. Task offloading to cloud or edge servers has also been studied, leveraging the ample resources of cloud servers for resource-constrained mobile systems and the ability of edge servers to efficiently meet deadline constraints with stable network connections. In this paper, we quantify the effectiveness of task offloading to cloud and edge servers in terms of power savings for mobile systems. Unlike previous approaches, we analyze the impact of offloading when co-optimized with other energy-saving techniques, such as dynamic voltage scaling and low-power memory configurations. Through extensive experiments, we explore the trade-offs between cloud and edge environments, accounting for workload intensity, network conditions, and server computing capabilities. Our findings offer valuable insights for designing optimized task offloading strategies tailored to specific mobile system characteristics, effectively balancing the benefits of cloud and edge environments.**

***Keywords* — real-time task, offloading, cloud, scheduling, edge, optimization**

I. INTRODUCTION

The advancements in Internet of Things (IoT) and Artificial Intelligence (AI) technologies have transformed various sectors, including healthcare [1, 2], manufacturing [3], transportation [4], and disaster management [5, 6]. As a result, there has been a rapid increase in the deployment of embedded systems that rely on real-time data acquisition, transmission, and AI-driven training and inference through sensors [7–10]. In such systems, minimizing energy consumption while adhering to the deadline constraints of real-time workloads is critical [11, 12]. To meet these demands, researchers have been focusing on optimizing the trade-off between performance and energy consumption across different system layers, dynamically adjusting these factors based on the workload intensity [13, 14].

For instance, when workload demand is low, resources can be configured to operate in low-power modes while still meeting deadline constraints, thereby reducing energy consumption [15]. Conversely, when workload demand is high, resource configurations are adjusted to maximize performance, even at the cost of increased energy consumption.

At the memory layer, techniques such as integrating low-power memory (LPM) into conventional DRAM have been studied. LPM (also known as NVM or persistent memory) consumes less power but offers lower performance than traditional DRAM [16, 17]. In these approaches, DRAM is activated for frequently accessed data, while LPM is prioritized for less active periods. For the processor layer, dynamic voltage and frequency scaling (DVFS) techniques adjust processor speeds based on computational demand — lowering speeds for light tasks and maximizing them for more demanding ones [18]. Additionally, task offloading to remote servers has been suggested as a solution for mobile systems with limited computing resources, especially when handling compute-intensive workloads [19]. Offloading to cloud servers, which have abundant computational resources, reduces the burden on mobile processors and leads to significant energy savings. Alternatively, offloading to nearby edge servers with stable network connections provides predictable latency, which is crucial for real-time systems while also achieving energy savings for mobile devices [20, 21].

In this paper, we analyze the impact of task offloading to cloud and edge servers on energy-saving optimization in mobile real-time systems. Specifically, we investigate the combined effects of processor's DVFS, low-power memory techniques, and task offloading to remote servers, comparing the effectiveness of cloud versus edge server offloading under various conditions. Our extensive experiments demonstrate that offloading to cloud servers is more efficient in terms of energy savings for high-computation workloads exceeding 50% of the full capacity of a mobile resource. In contrast, offloading to edge servers is more effective when the workload is below this threshold. However, the trade-offs between cloud and edge offloading are influenced by factors such as network bandwidth and the computational capabilities of the respective servers.

Specifically, when the capacity of a cloud server is no more than twice that of an edge server, offloading to the edge server proves to be more efficient.

The findings of this research provide valuable insights for designing optimized offloading strategies integrated with DVFS and low-power memory techniques in mobile embedded systems, taking into account the specific characteristics of real-time tasks and the diverse environmental factors of cloud and edge servers.

The remainder of this paper is structured as follows: Section II describes the task execution model used in this study. Section III describes the optimized execution of real-time tasks in our study. Section IV presents an analysis of how task offloading to cloud and edge servers varies depending on workload demand. Finally, Section V concludes the paper.

II. TASK EXECUTION MODEL

The task execution model in this paper extends the conventional real-time task model to include optimizations such as processor speed control, low-power memory allocation, and offloading to remote servers [22]. The set of tasks is defined as $T = \{t_1, t_2, \ldots, t_n\}$, and the target mobile system consists of a processor with Dynamic Voltage and Frequency Scaling (DVFS) capabilities, as well as main memory comprising low-power memory (LPM) and high-performance memory (HPM) in the form of DRAM. Additionally, we assume the availability of cloud and edge servers capable of handling offloaded tasks. Each task can be executed either on the local mobile processor or offloaded to a remote cloud or edge server, which leads to dividing the task set T into two subsets: `MOBILE` and `OFFLOADED`. Although increasing the offloading ratio is generally more efficient due to the superior computing power of remote servers, tasks must be carefully chosen for offloading. Tasks offloaded to remote servers must return their results before their deadlines, and some control tasks that interact with sensors or actuators should always be executed locally on the mobile processor.

Since tasks can reside in two different memory types, the task set T is further divided into `HPM` and `LPM`. A task t_i is defined as $t_i = <WCET_i, Period_i, Data_i>$, where $WCET_i$ represents the worst-case execution time of task t_i when executed at the default clock speed of the mobile processor. $Period_i$ denotes the execution period, and $Data_i$ represents the data characteristics, defined as $Data_i = <Size_i, Input_i, Output_i, Rd_i, Wr_i>$. $Size_i$ refers to the memory footprint of t_i, $Input_i$ is the input data required for execution, and $Output_i$ is the output data generated. Rd_i and Wr_i denote the number of read and write operations to memory, respectively. For tasks $t_i \in$ `OFFLOADED`, the input data size $Input_i$ must be transmitted to the remote server before execution, and after execution, the output data $Output_i$ must be received from the remote server. This paper deals with periodic real-time tasks, where each task's deadline is determined by its period $Period_i$. The overall period $Epoch_T$ for all tasks $t_i \in T$ is defined as the least common multiple of the individual periods.

Since we focus on real-time systems, even in cases of temporary network disconnection, the tasks should meet their deadlines. Therefore, the mobile processor must satisfy the following utilization test at its maximum clock speed without relying on remote servers:

$$Utilization = \sum_{t_i \in T} \frac{WCET_i}{Period_i} \leq 1 \quad (1)$$

If Equation (1) is satisfied, preemptive scheduling via the Earliest-Deadline-First (EDF) algorithm is possible for the given task set. Since we use DVFS, the worst-case execution time (WCET) of each task must be recalculated based on the processor's clock speed, and the following test should be satisfied:

$$Utilization = \sum_{t_i \in T} \frac{DVFS\,(WCET_i)}{Period_i} \leq 1 \quad (2)$$

Here, $DVFS\,(WCET_i)$ represents the worst-case execution time of task t_i after applying DVFS to the processor. Since the system employs two types of memory (HPM and LPM), the following test must also be satisfied:

$$Utilization = \sum_{t_i \in HPM} \frac{DVFS_{HPM}\,(WCET_i)}{Period_i} + \sum_{t_i \in LPM} \frac{DVFS_{LPM}\,(WCET_i)}{Period_i} \leq 1 \quad (3)$$

The terms $DVFS_{HPM}\,(WCET_i)$ and $DVFS_{LPM}\,(WCET_i)$ are calculated as:

$$DVFS_{HPM}\,(WCET_i) = DVFS(WCET_i) \quad (4)$$

$$DVFS_{LPM}\,(WCET_i) = DVFS(WCET_i) + \Delta_{Rd} * Rd_i + \Delta_{Wr} * Wr_i \quad (5)$$

where Δ_{Rd} and Δ_{Wr} represent the read/write delay differences between HPM and LPM, respectively. Since the model supports task offloading to remote servers, the processor utilization must also satisfy the following equation:

$$Utilization = \sum_{t_i \in MOBILE \cap HPM} \frac{DVFS_{HPM}\,(WCET_i)}{Period_i} + \sum_{t_i \in MOBLIE \cap LPM} \frac{DVFS_{LPM}\,(WCET_i)}{Period_i} + \sum_{t_i \in OFFLOADED} \frac{Comnd_i}{Epoch_T} \leq 1 \quad (6)$$

Here, $Comnd_i$ represents the time required for the mobile processor to issue offloading commands to the network module for task t_i at the start of each $Epoch_T$. After determining the execution location and processor speed for each task, the mobile

Algorithm 1

encoding processor, memory, offloading information with 3 strings;
`initialize` (population);
while population does not converge
  `select` parents p1, p2 from population;
  offspring ← `crossover` (p1, p2);
  offspring ← `mutation` (offspring);
  `replace` population with offspring;
end while
return the best solution in population;

processor initiates the offloading commands for tasks in the `OFFLOADED` set at the beginning of $Epoch_T$, maximizing the parallelism between tasks executed on the mobile device and those on the server.

To ensure that the offloaded tasks meet their deadlines, the following condition must be satisfied:

$$\forall t_i \in \texttt{OFFLOADED},\ Trnarnd_i < Period_i \tag{7}$$

where $Trnarnd_i$ is calculated as:

$$Trnarnd_i = Comnd_i + Up_i + Remote\,(WCET_i) + Dn_i \tag{8}$$

where $Remote\ (WCET_i)$ denotes the worst-case execution time on the remote server, which is determined by the difference in clock frequency between the mobile and remote processors.

The upload time Up_i and download time Dn_i are defined as:

$$Up_i = \begin{cases} \dfrac{Size_i}{BW_{up}} & \text{if first execution} \\ \dfrac{Input_i}{BW_{up}} & \text{otherwise} \end{cases} \tag{9}$$

$$Dn_i = \frac{Output_i}{BW_{dn}} \tag{10}$$

where BW_{up} and BW_{dn} represent the uplink and downlink bandwidths, respectively.

The basic model aforementioned assumes a single-core processor of mobile systems. For multi-core processors, the right-hand side of Equations (1), (2), (3), and (6) can be adjusted by replacing 1 with the number of cores. In this case, multi-core scheduling can be performed using a P-Fair class of algorithms, instead of EDF [23]. Figure 1 illustrates the basic structure of the task execution model proposed in this paper.

III. Optimizaed Executions of Task Set

The objective of this study is to quantitatively analyze the energy-saving effects of task offloading to cloud versus edge servers. Rather than identifying the best offloading option for fixed conditions, we aim to explore a range of solutions that minimize power consumption under varying workload intensities, network conditions, and server capabilities. These

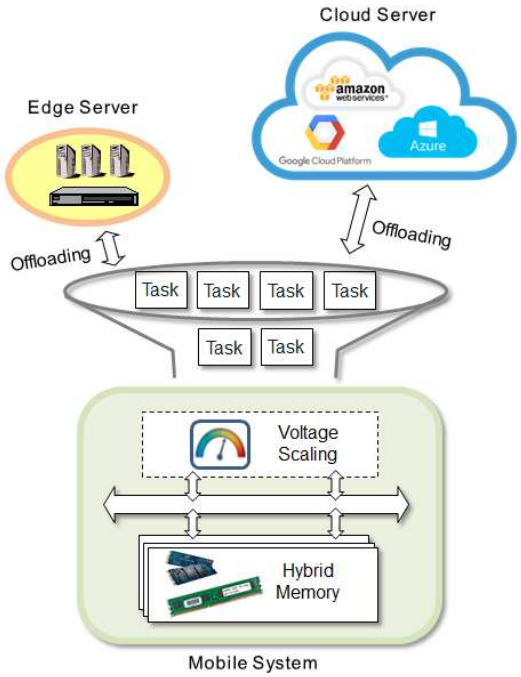


Fig. 1 Basic structure of the proposed task execution model.

factors may influence the energy-saving optimization of processor, memory, and other component configurations in mobile devices, which are comprehensively considered to evaluate the offloading effectiveness between cloud and edge.

In this paper, we use a genetic algorithm to optimize the execution of a given task set, by determining task location, processor clock frequency, and memory placement. A genetic algorithm is a probabilistic optimization method that mimics the principles of natural evolution in population genetics [24]. Our objective function aims to minimize the energy consumption of the mobile system, while ensuring that all tasks meet their deadlines. Tasks can be offloaded to either cloud servers or edge servers, and the mobile processor's clock frequency is defined in four levels: {0.125, 0.25, 0.5, 1.0}, where 1.0 represents the maximum clock frequency of the processor. Memory locations are represented as {0, 1}, indicating HPM (High-Performance Memory) or LPM (Low-Power Memory).

The solutions in the genetic algorithm consist of three strings per task, specifying the task's execution location (whether on the mobile device, edge server, or cloud server), the processor's clock frequency (chosen from the four defined levels), and the memory location (either HPM or LPM). The length of each string corresponds to the total number of tasks. The objective function used to evaluate the fitness of each solution is the energy consumption of the mobile device when scheduled with the given settings. If the processor's utilization exceeds 1 or the offloaded tasks fail the deadline test, a penalty is applied to the solution to promote the elimination of such attributes.

The population size for each generation in the genetic algorithm is set to 100, with the initial population generated randomly. After the initial population is created, a pair of parent solutions is selected in each generation, followed by crossover

and mutation operations, to generate new solutions [25]. This process is repeated until a converged set of solutions is achieved.

In the selection operation, solutions with better objective function values have a higher probability of being chosen as parents. Specifically, the selection probabilities are normalized based on the objective function values, ensuring that the best-ranked solution has four times the probability of being selected compared to the 100th-ranked solution. For the crossover operation, we use a one-point crossover, one of the most common techniques in genetic algorithms, where the segments on either side of a randomly chosen crossover point are inherited from different parent solutions [24]. To explore a broader search space, we apply the mutation operation after the crossover to perturb certain values in the solution [24]. The resulting new solutions are inserted into the next generation, while the weakest solutions are eliminated from the population.

The evolution process is repeated until the population converges. In our experiments, the genetic algorithm converged within an average of one second, confirming that the overhead is minimal. The experimental parameters followed those used in previous studies [22]. Algorithm 1 shows the pseudocode of the genetic algorithm used in this paper.

IV. Analyzing Effectiveness of Edge and Cloud Offloading

The objective function of this paper is to minimize the energy consumption of the mobile device. Energy consumption is measured by calculating the energy usage of the processor, memory, and network resources separately and then summing them. First, the processor energy $Energy_{\text{processor}}$ is defined as:

$$Energy_{\text{proceessor}} = c\, V_i^2\, f_i \left\{ \sum_{t_i \in MOBILE} DVFS\,(WCET_i) + \sum_{t_i \in OFFLOADED} Comnd_i \right\} \quad (11)$$

where c is the switching capacitance, V_i is the supplied voltage during the execution of task t_i, and f_i is the clock frequency. The network energy $Energy_{\text{network}}$ is defined as:

$$Energy_{\text{network}} = \sum_{t_i \in OFFLOADED} Net\_Power * (Up_i + Dn_i) \quad (12)$$

where *Net_Power* represents the power consumption of the network module. Memory energy is the sum of dynamic energy $Energy_{\text{memory\_d}}$ and static energy $Energy_{\text{memory\_s}}$, defined as follows:

$$Energy_{\text{memory\_d}} = \sum_{t_i \in HPM} (rd_i * HPM\_E_{rd} + wr_i * HPM\_E_{wr}) + \sum_{t_i \in LPM} (rd_i * LPM\_E_{rd} + wr_i * LPM\_E_{wr}) \quad (13)$$

$$Energy_{\text{memory\_s}} = \sum_{t_i \in HPM} (HPM\_Power * Size_i * T) + \sum_{t_i \in LPM} (LPM\_Power * Size_i * T) \quad (14)$$

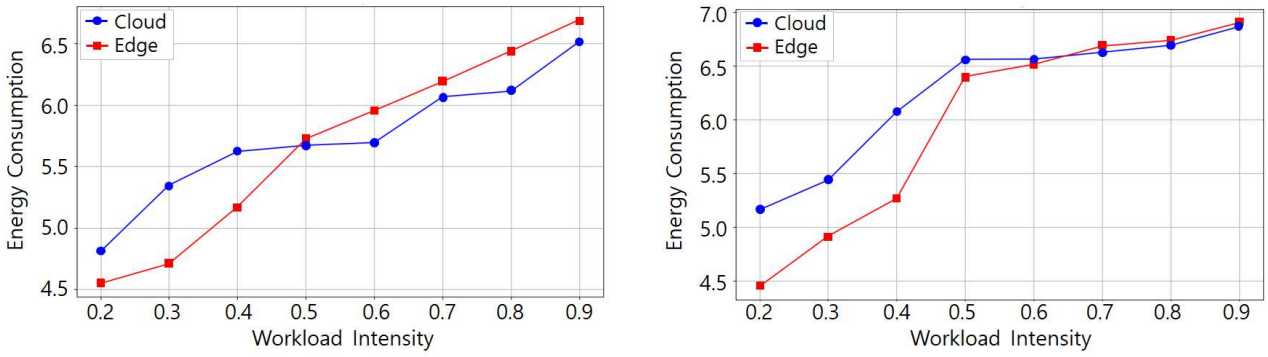
Here, HPM_E_{rd} and HPM_E_{wr} represent the energy consumption for reading and writing per access unit in HPM, while LPM_E_{rd} and LPM_E_{wr} represent the corresponding values for LPM. HPM_Power and LPM_Power refer to the power consumptions per unit capacity of HPM and LPM, respectively, and T is the total elapsed time while executing the task set.

In the experiment, both HPM and LPM are assumed to have sufficient capacity to store the entire task set, and tasks placed in LPM ensure that the corresponding HPM sections operate in low-power mode, thereby preventing static energy consumption. The access times for HPM and LPM are set to 50 nanoseconds and 100 nanoseconds for reads, and 50 nanoseconds and 350 nanoseconds for writes, respectively, based on previous studies [22]. The energy consumption for HPM and LPM is set to 0.1 nanojoule/bit and 0.2 nanojoule/bit for reads, and 0.1 nanojoule/bit and 1.0 nanojoule/bit for writes, respectively. The static power consumption values for HPM and LPM are set to 1 watt/GB and 0.1 watt/GB, respectively [22].

The task configurations are based on representative values from previous research [22]. The task set contains 100 tasks, and the worst-case execution time (WCET) is randomly assigned between 500 milliseconds and 1000 milliseconds. The task periods are set based on the determined WCET to match the load of the task set. The task sizes range from 500 kilobytes to 750 kilobytes, input sizes from 100 kilobytes to 500 kilobytes, and output sizes from 100 kilobytes to 250 kilobytes.

In our experiments, we evaluate energy consumption in mobile systems as the workload is offloaded to either edge servers or cloud servers, depending on task load variations. The task load ranges from a mobile processor utilization of 0.2 to 0.9, where a utilization of 1.0 represents the maximum processing capability of the mobile processor when operating at full speed.

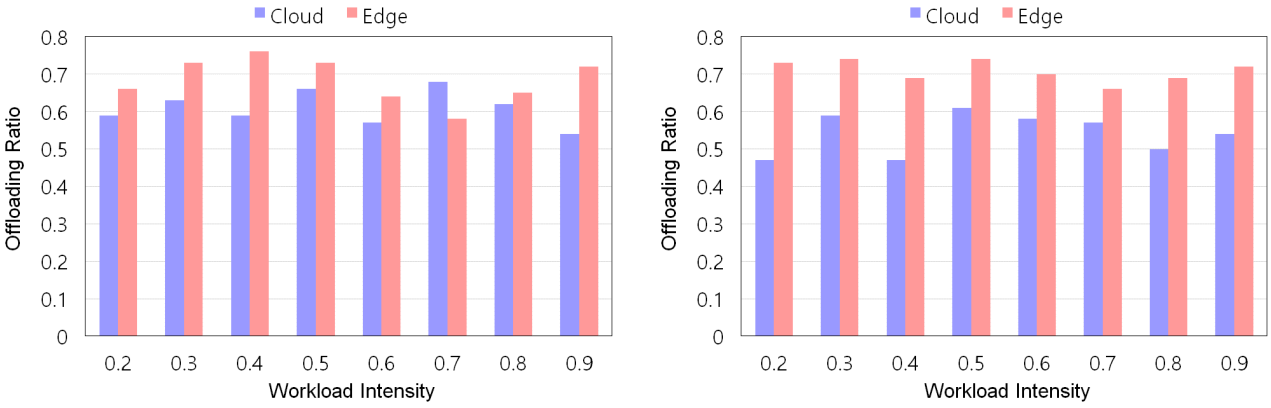
Figure 2(a) shows the energy consumption when offloading tasks to cloud and edge servers, with the cloud server having four times the computational capacity of the edge server. In this experiment, the network bandwidth for the edge server is set to 80 Mbps, as used in previous research [22], while the cloud server's bandwidth is set to 50 Mbps. As illustrated, under high workload intensity conditions, offloading to the cloud results in greater energy savings, demonstrating the advantage of the cloud server's superior computational capabilities in handling heavy loads. However, when the workload intensity is below 0.5, offloading to the edge server shows better energy-saving performance. Although the edge server has less computational power compared to the cloud server, the superior network bandwidth allows the edge server to achieve satisfactory results under low workload conditions.



(a) Cloud server with 4x edge server computing power

(b) Cloud server with 2x edge server computing power

Fig. 2 Comparison of mobile system's energy consumption when offloading to cloud and edge servers.



(a) Cloud server with 4x edge server computing power

(b) Cloud server with 2x edge server computing power

Fig. 3 Comparison of the optimized offloading ratio as the workload intensity is varied.

Figure 2(b) presents the energy consumption difference when the cloud server's computational capacity is reduced to twice that of the edge server. In this experiment, the cloud server's network bandwidth is set to 40 Mbps. The results indicate that, in most cases, offloading to the edge server results in better energy savings compared to the cloud server. Even when the workload exceeds 0.7, the cloud server shows only marginally better results, suggesting that when the cloud server's computational capacity is not vastly superior, the edge server's proximity and better network bandwidth provide more efficient scheduling results in mobile real-time systems.

Figure 3(a) compares the optimized offloading ratios of cloud and edge servers under the same conditions as Figure 2(a), where the cloud server's computational capacity is four times that of the edge server. As shown, the offloading ratio to the edge server is generally higher, although certain workload segments favor offloading to the cloud server. In both cases, the offloading ratio is consistently high, indicating that the computational power of remote servers is sufficiently large to offset the cost of offloading tasks via the network.

Figure 3(b) illustrates the optimized offloading ratio results in the environment of Figure 2(b), where the cloud server's computational capacity is reduced to twice that of the edge server, and its network bandwidth is also lowered. As shown, the offloading ratio to the cloud server is significantly lower than to the edge server in all cases, with this trend becoming more pronounced as the workload intensity decreases. This suggests that when the cloud server's computational capacity is only moderately better than the edge server's and network conditions are inferior, the relative benefit of offloading to the cloud diminishes. This effect becomes especially apparent under low workload intensity conditions, indicating that for cloud offloading to be effective, the workload intensity should be high, the network conditions favorable, and the computational capacity significantly superior.

V. Conclusion

In this paper, we designed a real-time task execution model that co-optimizes the energy-efficient configurations of resources in both mobile systems and remote servers. We then conducted extensive experiments to quantify the effectiveness

of edge/cloud offloading. Our experimental results indicate that offloading to cloud servers is more energy-efficient under high task load conditions exceeding 50% of a mobile processor's capacity, while offloading to edge servers yields better energy savings when the task load remains below this threshold. Moreover, the trade-offs between cloud and edge offloading are significantly influenced by factors such as network bandwidth and the computational capabilities of the respective servers. Specifically, when the capacity of a cloud server is no more than twice that of an edge server, offloading to the edge server proves to be a more effective approach. Based on these findings, we anticipate that optimized task offloading strategies can be realized by accounting for the specific characteristics of cloud and edge servers, the configurations of mobile resources such as processors and memory, and the unique properties of the real-time task set.

ACKNOWLEDGMENT

This work was supported in part by the National Research Foundation of Korea (NRF) under Grant RS-2024-00461678 and in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) under grant RS-2024-00459026 funded by Korean Government (MSIT).

REFERENCES

[1] F. Alshehri and G. Muhammad, "A Comprehensive Survey of the Internet of Things (IoT) and AI-Based Smart Healthcare," IEEE Access, vol. 9, pp. 3660-3678, 2021, doi: 10.1109/ACCESS.2020.3047960.

[2] N. Taimoor and S. Rehman, "Reliable and Resilient AI and IoT-Based Personalised Healthcare Services: A Survey," IEEE Access, vol. 10, pp. 535-563, 2022, doi: 10.1109/ACCESS.2021.3137364.

[3] V. Kharchenko, O. Illiashenko, O. Morozova and S. Sokolov, "Combination of Digital Twin and Artificial Intelligence in Manufacturing Using Industrial IoT," Proc. 11th IEEE Int'l Conf. on Dependable Systems, Services and Technologies (DESSERT), pp. 196-201, 2020, doi: 10.1109/DESSERT50317.2020.9125038.

[4] S. Chavhan, D. Gupta, S. Gochhayat, Chandana B. N., A. Khanna, K. Shankar, and J. Rodrigues, "Edge Computing AI-IoT Integrated Energy-efficient Intelligent Transportation System for Smart Cities," ACM Transactions on Internet Technology, vol. 22, no. 4, article 106, pp.1-18, 2022, doi: 10.1145/3507906.

[5] M. Lee and T. Chien, "Artificial Intelligence and Internet of Things for Robotic Disaster Response," Proc. IEEE Int'l Conf. on Advanced Robotics and Intelligent Systems (ARIS), pp. 1-6, 2020, doi: 10.1109/ARIS50834.2020.9205794.

[6] M. Abdalzaher, M. Krichen, and F. Falcone, "Emerging Technologies and Supporting Tools for Earthquake Disaster Management: A Perspective, Challenges, and Future Directions," Progress in Disaster Science, vol. 23, article 100347, pp. 1-28, 2024, doi: 10.1016/j.pdisas.2024.100347.

[7] J. Lee and H. Bahn, "File Access Characteristics of Deep Learning Workloads and Cache-Friendly Data Management," Proc. 10th IEEE Int'l Conf. on Electrical Engineering, Computer Science and Informatics (EECSI), pp. 328-331, 2023, doi: 10.1109/EECSI59885.2023.10295817.

[8] Y. Chen, B. Zheng, Z. Zhang, Q. Wang, C. Shen, and Q. Zhang, "Deep Learning on Mobile and Embedded Devices: State-of-the-art, Challenges, and Future Directions," ACM Computing Surveys, vol. 53, no. 4, pp. 1-37, 2021, doi: 10.1145/3398209.

[9] S. Kwon and H. Bahn, "Memory Reference Analysis and Implications for Executing AI Workloads in Mobile Systems," Proc. IEEE Int'l Conf. on Electrical and Information Technology (IEIT), pp. 281-285, 2023, doi: 10.1109/IEIT59852.2023.10335577.

[10] S. Nam and H. Bahn, "Adaptive Swapping for Variable Workloads in Real-time Task Scheduling," Proc. IEEE Int'l Conf. on Communications, Computing, Cybersecurity, and Informatics (CCCI), pp. 1-6, 2023, doi: 10.1109/CCCI58712.2023.10290800.

[11] J. Bi, H. Yuan, S. Duanmu, M. Zhou, and A. Abusorrah, "Energy-Optimized Partial Computation Offloading in Mobile-Edge Computing with Genetic Simulated-Annealing-based Particle Swarm Optimization," IEEE Internet of Things Journal, vol. 8, no. 5, pp. 3774-3785, 2021, doi: 10.1109/JIOT.2020.3024223.

[12] S. Panda, M. Lin and T. Zhou, "Energy-Efficient Computation Offloading With DVFS Using Deep Reinforcement Learning for Time-Critical IoT Applications in Edge Computing," IEEE Internet of Things Journal, vol. 10, no. 8, pp. 6611-6621, 2023, doi: 10.1109/JIOT.2022.3153399.

[13] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge Computing: Partial Computation Offloading Using Dynamic Voltage Scaling," IEEE Transactions on Communications, vol. 64, no. 10, pp. 4268-4282, 2016, doi: 10.1109/TCOMM.2016.2599530.

[14] R. Duan, J. Wang, C. Jiang, Y. Ren, and L. Hanzo, ''The Transmit-Energy vs Computation-Delay Trade-off in Gateway-Selection for Heterogeneous Cloud Aided Multi-UAV Systems,'' IEEE Transactions on Communications, vol. 67, no. 4, pp. 3026-3039, 2019, doi: 10.1109/TCOMM.2018.2889672.

[15] S. A. Nam, K. Cho, and H. Bahn, "A New Resource Configuring Scheme for Variable Workload in IoT Systems," Proc. IEEE Asia-Pacific Conf. on Computer Science and Data Engineering (CSDE), pp. 1-6, 2022, doi: 10.1109/CSDE56538.2022.10089270.

[16] E. Lee, H. Bahn, S. Yoo and S. H. Noh, "Empirical Study of NVM Storage: An Operating System's Perspective and Implications," Proc. 22nd IEEE Int'l Symp. on Modelling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 405-410, 2014, doi: 10.1109/MASCOTS.2014.56.

[17] J. Lee and H. Bahn, "Analyzing Data Access Characteristics of Deep Learning Workloads and Implications," Proc. 3rd IEEE Int'l Conf. on Electronic Information Engineering and Computer Science (EIECS), pp. 546-551, 2023, doi: 10.1109/EIECS59936.2023.10435537.

[18] S. Li, W. Sun, Y. Sun, and Y. Huo, "Energy-Efficient Task Offloading Using Dynamic Voltage Scaling in Mobile Edge Computing," IEEE Transactions on Network Science and Engineering, vol. 8, no. 1, pp. 588-598, 2021, doi: 10.1109/TNSE.2020.3046014.

[19] S. Raza, S. Wang, M. Ahmed, M. R. Anwar, M. A. Mirza, and W. U. Khan, "Task offloading and resource allocation for IoV using 5G NR-V2X communication," IEEE Internet of Things Journal, vol. 9, no. 13, pp. 10397-10410, 2022, doi: 10.1109/JIOT.2021.3121796.

[20] T. Zheng, J. Wan, J. Zhang, C. Jiang, and G. Jia, "A Survey of Computation Offloading in Edge Computing," Proc. IEEE Int'l Conf. on Computer, Information and Telecommunication Systems (CITS), pp. 1-6, 2020, doi: 10.1109/CITS49457.2020.9232457.

[21] S. Park and H. Bahn, "Trace-Based Performance Analysis for Deep Learning in Edge Container Environments," Proc. 8th IEEE Int'l Conf. on Fog and Mobile Edge Computing (FMEC), pp. 87-92, 2023, doi: 10.1109/FMEC59375.2023.10306027.

[22] S. Ki, G. Byun, K. Cho and H. Bahn, "Co-optimizing CPU voltage, memory placement, and task offloading for energy-efficient mobile systems," IEEE Internet of Things Journal, vol. 10, no. 10, pp. 9177-9192, 2023, doi: 10.1109/JIOT.2022.3233830.

[23] J. Anderson and A. Srinivasan, "Mixed Pfair/Erfair Scheduling of Asynchronous Periodic Tasks," Journal of Computer and System Sciences, vol. 68, pp. 157-204, 2004, doi: 10.1016/j.jcss.2003.08.002.

[24] D. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Boston, MA, 1989.

[25] D. Whitley and T. Starkweather, "Genitor II: A Distributed Genetic Algorithm," Journal of Experimental & Theoretical Artificial Intelligence, vol. 2, no. 3, pp. 189-214, 1990, doi: 10.1080/09528139008953723.