Contents lists available at Science-Gate

International Journal of Advanced and Applied Sciences

Journal homepage: http://www.science-gate.com/IJAAS.html

A review of parallel processing in resource-constrained Internet of Things (IoT) devices

Nasser S. Albalawi [1,*], Abdulaziz Ghabash Alanazi [2], Sami Alshammari [3], Fahd Alhamazani [1], Amnah A. Alshammari [2]

[1]*Department of Computer Science, Faculty of Computing and Information Technology, Northern Border University, Rafha, Saudi Arabia*
[2]*Department of Information Systems, Faculty of Computing and Information Technology, Northern Border University, Rafha, Saudi Arabia*
[3]*Department of Information Technology, Faculty of Computing and Information Technology, Northern Border University, Rafha, Saudi Arabia*

ARTICLE INFO

Article history:
Received 10 February 2025
Received in revised form
24 June 2025
Accepted 5 August 2025

Keywords:
Internet of Things
Parallel processing
Resource constraints
IoT efficiency
Embedded systems

ABSTRACT

The Internet of Things (IoT) has transformed the connection between physical and digital systems by enabling continuous data exchange and communication. However, the rapid increase in IoT devices brings significant challenges due to limited memory, processing power, and low-energy communication standards. Addressing these resource constraints is essential for improving system performance. This review explores existing parallel processing techniques specifically developed for resource-limited IoT devices, including hardware and software approaches that aim to enhance efficiency and speed. A comprehensive analysis of the literature highlights the importance of parallel processing in overcoming these limitations. The paper also discusses key challenges, potential benefits, and future directions, aiming to guide further research toward more efficient use of computational resources in IoT environments.

1. Introduction

The Internet of Things (IoT), a term introduced by Kevin Ashton in 1999 (Ashton, 2009), describes a network of physical devices equipped with sensors, software, and related technologies that enable data collection and exchange (Chander and Kumaravelan, 2019). While the idea was originally associated with RFID tags, it has since expanded to include a wide variety of distinct devices capable of autonomous communication. Through their integration, these smart devices form an intelligent system in which continuous data sharing improves performance and efficiency. As a major technological development, IoT extends the traditional Internet into a new computing model, supporting widespread connectivity and intelligent interaction among diverse devices (Atzori et al., 2017). IoT has received substantial recognition in recent years because of its great potential across multiple sectors, notably automated transport, logistics, homes, cities, healthcare, environmental monitoring, infrastructure, Industry 4.0, and agriculture (Raj et al., 2021; Malik et al., 2021; Bibri, 2018; Krishnamoorthy et al., 2023). The devices are important to every IoT solution, as they are equipped with sensors, processors, and actuators that sense, gather, transmit, process, and react to input. These devices communicate with other devices, networks, and services based on data regulation and oversight, improving the effectiveness and performance of many enterprises and activities. IoT devices may exchange data and information with other computing systems and with one another, allowing for the monitoring, control, and optimization of activities, including energy management, manufacturing, logistics, and transportation (Ahmad and Zhang, 2021; Nižetić et al., 2020).

IoT devices enable automated decision-making by combining data from many sources, further increasing production and effectiveness across various industries. IoT devices can be categorized into two types: resource-abundant devices, such as servers, PCs, tablets, and cellphones, and resource-scarce devices, including industrial sensors, RFID

* Corresponding Author.
Email Address: nasser.albalawi@nbu.edu.sa (N. S. Albalawi)
https://doi.org/10.21833/ijaas.2025.09.003
Corresponding author's ORCID profile:
https://orcid.org/0000-0002-5948-4260

tags, and actuators (Fig. 1). The proliferation of IoT is expected to significantly impact the market by enhancing the efficiency of data exchange between all parties involved. As IoT technology becomes increasingly prevalent, it will lead to a more effective and seamless data transfer process across various sectors, optimizing communication and operational workflows.

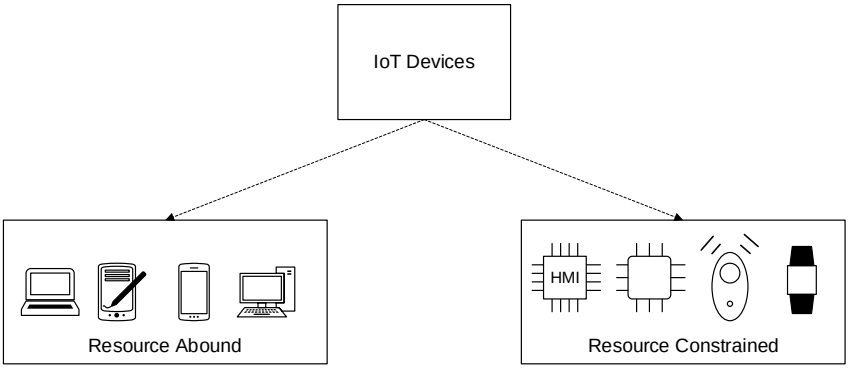

Fig. 1: Two major kinds of IoT devices (Deep et al., 2022)

However, by design, IoT devices are constrained by limited computational power, memory, energy, and bandwidth, presenting significant challenges for executing complex tasks. These devices often operate on low-power processors and have restricted memory capacities to maintain a small form factor and cost-effectiveness, limiting their ability to handle intensive computational processes. Due to reliance on batteries or energy-harvesting methods, energy constraints necessitate efficient power management strategies to prolong operational lifespan and maintain functionality in remote or inaccessible locations (Ijemaru et al., 2022; Sanislav et al., 2021). Additionally, limited bandwidth hampers the ability to transmit large volumes of data, requiring optimized communication protocols to ensure data integrity and timely transmission (Makhdoom et al., 2019). Addressing these challenges necessitates innovative approaches such as edge computing, where data processing is offloaded to nearby edge servers, reducing the computational burden on individual devices.

In this case, parallel processing emerges as a potent solution to address these challenges. By distributing computational tasks across multiple processing units, parallel processing can significantly improve the efficiency and speed of operations in resource-constrained IoT devices (Shuvo et al., 2022). This approach not only accelerates data processing but also optimizes energy consumption, which is crucial for the prolonged operation of battery-powered IoT devices. The integration of parallel processing in IoT devices requires careful consideration of various factors. These include the architectural design of the devices, the nature of the tasks to be parallelized, and the communication overhead between processing units. Furthermore, strategies such as lightweight parallel algorithms, energy-efficient scheduling, and task offloading to more capable edge or cloud servers play a critical role in enhancing the capabilities of IoT devices.

Despite significant advancements in parallel processing techniques for resource-constrained IoT devices, there remains a critical research gap in optimizing energy efficiency while maintaining high performance. Current literature predominantly focuses on enhancing computational speed and task distribution across multiple nodes; however, these approaches often neglect the substantial energy overhead associated with parallel processing in constrained environments. Moreover, the existing studies primarily address theoretical models or simulations rather than real-world applications, which leads to a need for more practical, scalable solutions for diverse IoT deployments. Addressing this gap requires a holistic approach that integrates energy-aware parallel processing algorithms with adaptive power management strategies explicitly tailored for IoT devices with limited computational and energy resources.

The rest of the paper is structured as follows: Section 2 reviews the existing literature on resource-constrained IoT and parallel processing. Section 3 explores resource constraints and parallel processing techniques specific to IoT devices. Section 4 presents detailed case studies and practical applications. Section 5 discusses the challenges and limitations faced in this domain, while Section 6 outlines future research directions. Lastly, Section 7 offers concluding remarks and summarizes the key findings.

2. Literature review

2.1. IoT

An important step towards the development of IoT devices was the 1996 definition of IPv6, addressed by the Internet Engineering Task Force (IETF) (Deep et al., 2022). Meeting the ever-increasing demands of the modern internet has been made possible by this breakthrough, and others like IPv6 low power wireless personal area network (6LoWPAN), and IEEE 802.15.4 (Khattak et al., 2023; Al-Kashoash et al., 2019). Ziegler et al. (2015) claimed that IPv6, with its vastly expanded address space, provides the scalability needed to accommodate the billions of IoT devices expected to

come online. Additionally, IEEE 802.15.4 offers a reliable framework for low-rate wireless personal area networks, which are essential for IoT connectivity as exhibited by Fuentes-Samaniego et al. (2019). Meanwhile, 6LoWPAN enables IPv6 packets to be sent and received over IEEE 802.15.4 networks, bridging the gap between traditional IP networks and the resource-constrained networks typical of IoT environments. Together, these technologies form the backbone of the IoT, supporting a seamless and efficient network that can handle the complexity and scale of present-day Internet applications, as illustrated in Fig. 2.

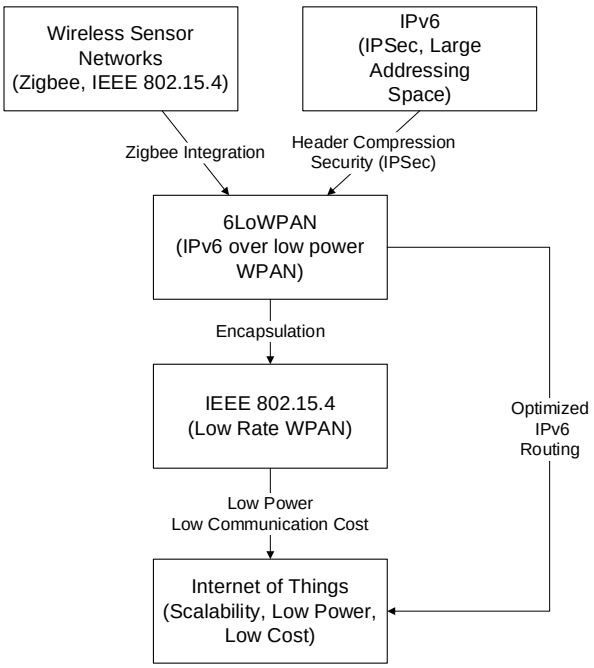

Fig. 2: Background of IoT (Deep et al., 2022)

However, the proliferation of IoT applications is revolutionizing industries, with enterprises increasingly recognizing their value. IoT will generate $14.4 trillion in value from 2013 to 2024, primarily through increased revenues and cost reductions. Manufacturing, retail trade, information services, and finance and insurance sectors are the primary beneficiaries, accounting for over half of this value. In the U.S., the services sector is poised to capture $4.6 trillion, while China's $1.8 trillion share is driven by its robust manufacturing growth.

Advancements in foundational IoT technologies, including networks, software, hardware, and data processing, underpin this transformation (Allioui and Mourdi, 2023). Networks are evolving towards unobtrusive, wire-free communication, enhancing device-to-device interactions. IoT software development is increasingly focused on interoperability, security, and distributed intelligence, exemplified by Google's $3.2 billion acquisition of Nest (Lee and Lee, 2015). Hardware innovations prioritize miniaturization and energy efficiency, while data processing techniques are becoming more context-aware and cognitive. These advances enable real-time analysis of vast data streams, which are crucial for applications like smart grids, environmental monitoring, and smart manufacturing. Optimized data processing is essential for making timely, informed decisions, underscoring its importance in the expanding IoT ecosystem.

The evolution of IoT technologies across network architecture, software, hardware, and data processing is presented in Table 1, showing significant milestones from before 2010 through projections beyond 2024.

Table 1: Advancement of IoT (Lee and Lee, 2015)

	Before 2010	2010-2015	2015-2024	Beyond 2024
Network	Sensor networks	Networks respond to their own needs Accuracy in monitoring network locations Networks that can withstand delays Utility and data storage networks Connectivity methods that combine	Network context awareness	Cognition in networks Networks that learn and fix themselves
Software and algorithms	Relational database integration IoT-oriented RDBMS Event-based platform Sensor middleware Sensor network middleware Proximity/Localized algorithms	Transparent massive software components Programs that can be composed Social technology built on the IoT Next-gen business apps built on IoT	Goal-oriented software Distributed intelligence, problem-solving Things-to-Things collaboration environments	Software tailored to users The invisible IoT Software that is easy to deploy Collaboration between things and humans The IoT for all
Hardware	RFID tags and some sensors Sensors are built into mobile devices NFC in mobile phones Smaller and cheaper MEMs technology	Readers that support many protocols and standards. More sensors and actuators Safe, inexpensive tags (like Silent Tags) An increase in the number of sensors and operators	Smart sensors (biochemical) More sensors and actuators (tiny sensors)	Nanotechnology and novel materials
Data processing	Serial data processing Parallel data processing Quality of services	Managing information while being mindful of energy, frequency, and spectrum Data processing setting adaptability	Context-aware data processing and data responses	Thinking things through and making improvements

2.2. Constrained devices

Embedded devices on the IoT ecosystem must possess both computational capabilities to perform their designated tasks and networking abilities to facilitate Internet integration. These devices, designed to be cost-effective, are often equipped with low-power embedded computational units, typically with limited storage and memory capacities. For instance, the RedBee EconoTAG, a representative low-power constrained device, offers 96 KB of RAM (Sehgal et al., 2012). However, the

execution model of this device requires that the contents of its flash memory (excluding the bootloader) be copied to RAM before execution, further reducing the available memory for data storage.

Given these stringent memory constraints, networking technologies for IoT devices must be designed with these limitations in mind (Amadeo et al., 2016). Therefore, identifying a minimal IP-based protocol set that can efficiently manage IoT devices is crucial. This protocol set must retain the essential features that ensure protocols remain recognizable and compatible with existing tools, thus enabling true interoperability across diverse devices. The challenge lies in balancing functionality with resource efficiency, ensuring that even with constrained memory and processing power, the devices can still communicate effectively within the IoT framework.

To achieve this, protocols such as 6LoWPAN and CoAP (Constrained Application Protocol) are often employed (Devasena, 2016). Fossati and Tschofenig (2016) outlined TLS and DTLS profiles specifically designed for secure communication in IoT environments. He explained that DTLS (Datagram Transport Layer Security) is used to provide secure communication over UDP, maintaining a lightweight footprint suitable for constrained devices. By focusing on these minimal yet essential protocol sets, IoT devices can achieve the necessary functionality and interoperability while operating within their limited resource environments. Table 2 provides a summary of a few common low-power limited devices.

Table 2: An overview of various devices with low power consumption (Sehgal et al., 2012)

Type	CPU	RAM	Flash/ROM
Crossbow TelosB	16-Bit MSP430	10 KB	48 KB
RedBee EconoTAG	32-Bit MC13224v	96 KB	128 KB
Atmel AVR Raven	8-Bit ATMega1284P	16 KB	128 KB
Crossbow Mica2	8-Bit ATMega 128L	4 KB	128 KB

2.3. Parallel processing

In the past decade, parallel computing has become a critical approach for optimizing performance in resource-constrained devices within the IoT sector. Unlike serial processing, which handles tasks sequentially, parallel computing divides tasks into smaller units that are processed simultaneously across multiple processing units, significantly reducing computation time, as illustrated in Kan et al. (2018). As shown in Fig. 3, Parallel computing using off-the-shelf methodologies like multi-threading and single-instruction-multiple-data (SIMD) has been made much easier with the introduction of multi-core CPUs, GPUs, and cloud computing infrastructure. An example of such a framework is the one created by Raghavan and Waghmare (2002) for use in manufacturing applications; it combines a work-stealing scheduler with a tree-structured model to dynamically manage worker participation and integrates different, restricted computer resources. Mourtzis et al. (2016) presented that cloud computing improves process planning in decentralized assembly lines through real-time monitoring and task assignment. To efficiently decompose services and choose the best ones, (Tao et al., 2012), a parallel computing technique that outperformed conventional serial algorithms was presented. In the IoT sector, Sehgal et al. (2012) paralleled processing was utilized to manage resource-constrained devices while creating a model-driven parallel processing system based on user-defined functions to address similar challenges. These advancements underscore the transformative impact of parallel computing in managing and optimizing complex tasks in constrained environments.

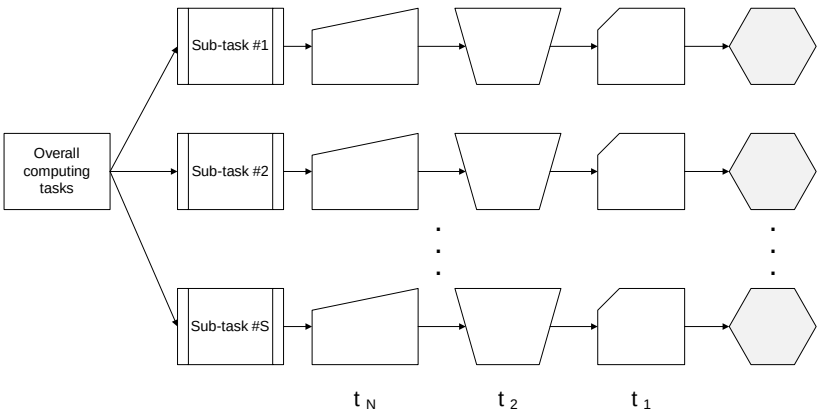

Fig. 3: Parallel computing across multiple processors (Sehgal et al., 2012)

Consequently, picking the correct system is crucial for implementing and testing a set of management for devices with limited resources. Parallel processing in the context of IoT for resource-constrained environments involves the simultaneous execution of multiple tasks to optimize performance and efficiency (Jeyaraj et al., 2023; Khalil et al., 2020). IoT devices, which frequently have

constrained memory, processing, and power resources, necessitate this strategy. By leveraging parallel processing, tasks such as data collection, processing, and transmission can be distributed across multiple nodes or cores, thus reducing latency and enhancing throughput. Efficient algorithms and protocols are designed to manage and synchronize these parallel tasks, ensuring the limited resources are utilized effectively without overwhelming the device. This not only improves the responsiveness and reliability of IoT systems but also extends the operational lifespan of devices by minimizing energy consumption and optimizing resource allocation.

3. Parallel processing fundamentals and concepts

3.1. Parallel processing architecture

In parallel processing, numerous computations or processes are executed concurrently, making use of the processing capability of numerous processors to resolve issues more effectively. This approach is essential in modern computing to handle complex tasks and large data sets efficiently. There are several architectures within parallel processing, each designed to optimize specific types of workloads. The primary architectures include SIMD (Single Instruction, Multiple Data), MIMD (Multiple Instruction, Multiple Data), and SPMD (Single Program, Multiple Data). Each of these architectures has distinct characteristics and applications.

3.1.1. SIMD

SIMD is a parallel computing architecture that excels in concurrently applying a single instruction to numerous data points. This design is particularly advantageous for tasks requiring uniform operations on extensive data sets, such as image processing, matrix multiplication, and scientific simulations. SIMD is integrated into various hardware implementations, including vector processors and modern graphics processing units (GPUs) (Zhang et al., 2022), making it highly versatile. The key characteristics of SIMD include its ability to perform identical operations on all elements of a data set simultaneously, thereby achieving substantial data parallelism. This architecture is highly efficient in vector and matrix operations, delivering high throughput by processing large volumes of data swiftly. Additionally, the inclusion of SIMD units in processors enhances hardware efficiency with minimal overhead, significantly boosting performance without considerable complexity increases.

3.1.2. MIMD

MIMD architecture represents a highly flexible and scalable approach to parallel computing, allowing multiple processors to execute different instructions on different data simultaneously. This architecture excels in handling a broad spectrum of applications, ranging from complex simulations to running various programs concurrently on multi-core processors. Key characteristics of MIMD include instruction parallelism, where different processors execute distinct instructions at the same time, and task parallelism, which is ideal for dividing tasks into independent subtasks, each requiring unique processing. The advantages of MIMD are notable: its flexibility accommodates complex, non-uniform workloads, while its scalability permits expansion by adding more processors, whether in a shared-memory configuration like multi-core processors or within a distributed system such as clusters. This makes MIMD architecture a powerful choice for modern computing environments demanding high performance and versatility (Yuan et al., 2013).

3.1.3. SPMD

SPMD is a model for parallel computing in which numerous processors run the same program on separate data sets. Predominantly utilized in distributed memory systems, SPMD is a subset of the MIMD model. It allows each processor to run identical code while operating on distinct data segments, which facilitates significant parallelism. This approach is particularly advantageous for large-scale numerical simulations and similar problems that can benefit from data partitioning. The uniformity in program execution simplifies the development and debugging processes compared to the more complex MIMD model. Consequently, SPMD offers a balance of simplicity and efficiency, making it a practical choice for achieving effective parallelism in various computational tasks.

3.2. Parallel processing framework

Parallel processing frameworks are essential for handling large-scale data processing tasks efficiently by distributing the workload across multiple computing nodes. Three prominent frameworks in this domain are MapReduce, Apache Spark, and Hadoop (Farhan et al., 2018). Each framework offers unique features and capabilities tailored to different types of data processing requirements.

3.2.1. MapReduce

To manage and generate massive data sets across vast clusters of computers, Google created the sophisticated programming paradigm and processing approach known as MapReduce. This approach breaks down distributed data processing into its two main components, Map and Reduce, which makes the process much easier to understand and implement. The input data is processed by the Map function, which then produces a set of intermediate key-value pairs. The final output is generated by merging all the intermediate values associated with the same key, which is done by the

Reduce function (Yang et al., 2007). Because of its famed scalability and fault tolerance, the MapReduce framework is ideal for processing data on a massive scale. Optimizing data distribution and parallel computation, MapReduce guarantees stability in the face of machine failures while efficiently processing massive amounts of data.

3.2.2. Apache Spark

Apache Spark is a powerful open-source unified analytics engine designed for large-scale data processing, enhancing the traditional MapReduce model to support a broader array of computations, including interactive queries and real-time stream processing. One of its standout features is in-memory computing, which significantly accelerates data processing tasks compared to disk-based alternatives, making it capable of executing batch-processing jobs up to 100 times faster than Hadoop MapReduce. Spark's ease of use is evident through its intuitive APIs available in Java, Scala, Python, and R, catering to a diverse group of developers. Additionally, Spark excels in advanced analytics, supporting complex operations such as machine learning, graph processing, and SQL queries (Ketu et al., 2020). This versatility and efficiency make Apache Spark a preferred choice for contemporary data engineering and analytics tasks, enabling organizations to handle diverse and large-scale data workloads with remarkable speed and simplicity.

3.2.3. Hadoop

Hadoop, developed by the Apache Software Foundation, is an open-source framework tailored for the distributed storage and processing of extensive datasets. It excels in scalability, seamlessly expanding from single servers to thousands of machines, each contributing local computation and storage capabilities. At its core, Hadoop comprises key components like the Hadoop Distributed File System (HDFS), which facilitates high-throughput data access by distributing large files across multiple machines (Ketu et al., 2020).

YARN (Yet Another Resource Negotiator) efficiently manages cluster resources, ensuring optimal allocation and scheduling of users' applications. Additionally, Hadoop pioneered the MapReduce model, enabling parallel processing of massive data sets. Beyond its foundational elements, Hadoop boasts a rich ecosystem encompassing tools such as Hive for SQL queries, Pig for high-level data flows, and HBase for NoSQL database functionality (Ketu et al., 2020). This comprehensive suite makes Hadoop a robust solution for organizations seeking to manage, process, and derive insights from big data on scale.

Fig. 4 shows a comparison between MapReduce tasks in Hadoop and the CEP operator chain used in Spark, demonstrating differences in data flow and processing logic (Kotenko et al., 2017).

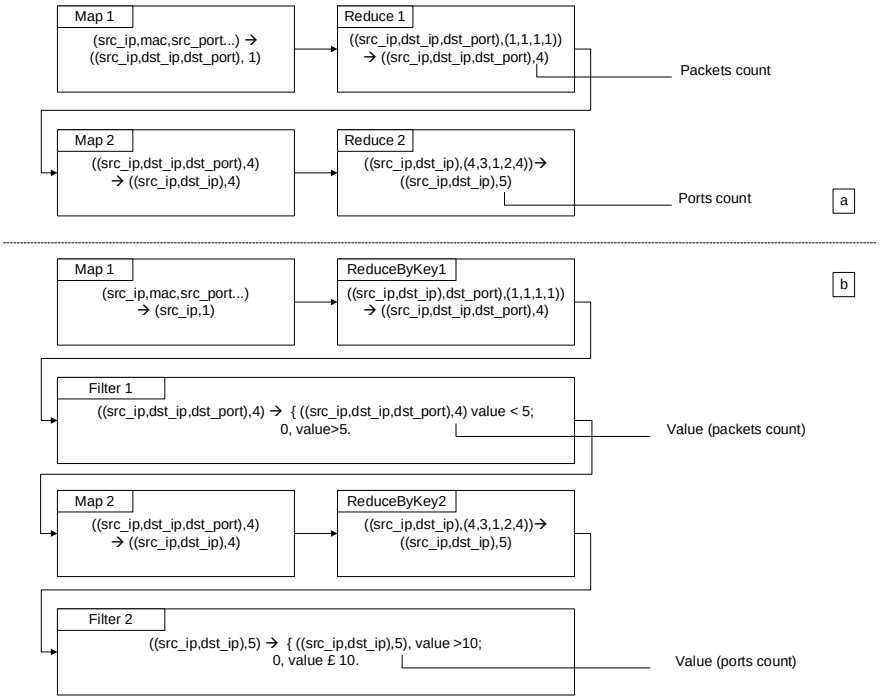

Fig. 4: (a) MapReduce tasks for Hadoop data analysis, (b) CEP operator chain for Spark data analysis

3.3. Parallel processing algorithms

3.3.1. Parallel sorting algorithms

Parallel sorting algorithms are tailored to efficiently handle the sorting of large datasets by leveraging concurrent processing across multiple processors or cores. Examples of such algorithms include Parallel Quicksort, which divides the array into smaller sub-arrays that are sorted concurrently and then merged to achieve the final sorted result (Amrahov et al., 2024). Another method, Parallel Mergesort, partitions the array into segments that are independently sorted in parallel and

subsequently merged into a fully sorted array. Additionally, Parallel Bitonic Sort employs a divide-and-conquer approach, utilizing a specific sequence of comparisons to sort elements in parallel (Liyanage, 2017). These algorithms optimize sorting performance by distributing the computational load, thereby harnessing the capabilities of modern parallel computing architectures effectively.

3.3.2. Parallel search algorithms

Parallel search algorithms leverage the power of parallel processing to efficiently locate specific elements within datasets or graph structures. Among these algorithms, Parallel Binary Search stands out for its ability to distribute the search operation across multiple processors, employing binary search principles to swiftly pinpoint the target element. In contrast, Parallel Depth-First Search (DFS) harnesses parallelism to explore multiple paths concurrently within graph or tree structures, facilitating effective traversal and element discovery. Similarly, Parallel Breadth-First Search (BFS) operates by exploring nodes or elements level by level in parallel, ensuring comprehensive coverage of the graph or tree while optimally utilizing the computational resources (Liu and Huang, 2015). These algorithms exemplify the versatility and efficiency achieved through parallel processing techniques in the realm of search operations, catering to diverse application scenarios where speed and scalability are paramount.

3.3.3. Parallel machine learning algorithms

Parallel machine learning algorithms harness the power of distributed computing to expedite model training on extensive datasets. For instance, Parallel Gradient Descent distributes gradient computations across multiple processors, enabling simultaneous optimization of model parameters (Kennedy et al., 2019). Similarly, Parallel Random Forest trains decision trees concurrently across multiple processors, amalgamating their outputs to achieve heightened accuracy. In the realm of deep learning, Parallel Neural Networks leverage parallel processing for batch training and inference, significantly enhancing the efficiency of training large-scale models (Kahira et al., 2021). These approaches not only accelerate computation but also facilitate handling vast amounts of data, making them pivotal in modern machine-learning applications where speed and scalability are paramount. Fig. 5 presents the overall concept of parallel processing using Hadoop and Spark, showing how data is collected, stored, and visualized for security analysis (Kotenko et al., 2017).

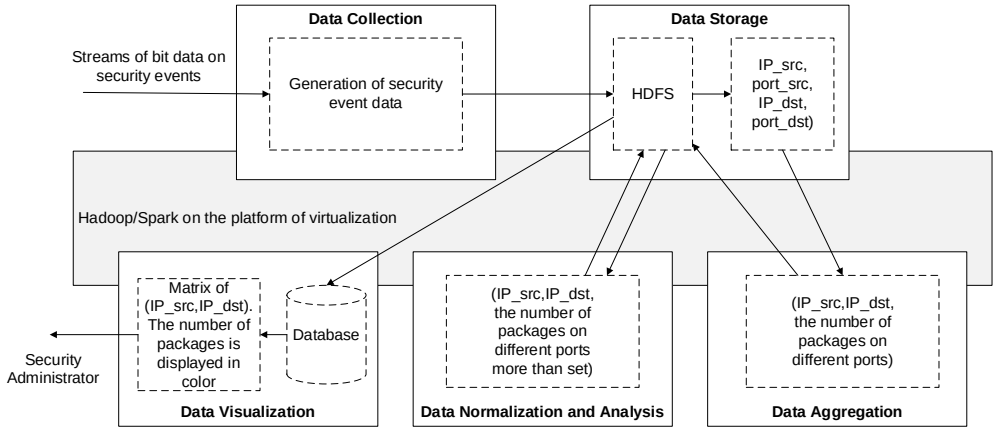

Fig. 5: Concept of parallel processing

3.4. Comparative evaluation of parallel processing methods in IoT

Existing parallel processing methods for IoT devices vary in architecture, efficiency, and suitability for constrained environments. Among the widely adopted paradigms are task parallelism, data parallelism, and model parallelism. Task parallelism divides the application into independent subtasks that run concurrently, often yielding high flexibility but requiring careful task scheduling to avoid load imbalance.

Data parallelism, in contrast, is highly scalable and suitable for sensor data processing and real-time analytics but may incur significant communication overhead if data distribution is not optimized. The comparison is shown in Table 3.

Table 3: Trade-offs of parallel processing methods in IoT

Method/framework	Strengths	Limitations	Suitability for IoT
Task parallelism	Flexible; simple logic separation	Load imbalance; scheduling complexity	Moderate
Data parallelism	High scalability; efficient for large data streams	Communication overhead; data partitioning complexity	High
Model parallelism	Enables large ML models in limited memory	Synchronization issues; inter-device latency	Moderate to High (Edge ML)
MapReduce	Fault-tolerant; reliable for batch jobs	High latency; disk-based processing	Low (edge/real-time apps)
Apache Spark	In-memory speed; rich analytics	Memory-intensive; not suitable for low-power devices	Moderate to High (Edge/Cloud)
GPU acceleration	High throughput for parallel tasks	Power hungry; hardware dependency	Low to Moderate
Cloud offloading	Vast resources; offloads computation from the device	Latency; data privacy concerns	High (non-real-time IoT)

3.5. Parallel processing techniques for IoT devices

In the realm of IoT, where devices are often resource-constrained yet tasked with handling large amounts of data and complex computations, parallel processing techniques play a pivotal role in optimizing performance and efficiency. This section delves into several key methodologies employed in parallel processing for IoT devices.

3.5.1. Task partitioning and offloading

Task partitioning and offloading are crucial strategies in optimizing computational tasks across heterogeneous IoT devices. In IoT environments, where devices vary significantly in processing capabilities and memory, task partitioning enables efficient utilization of resources by breaking down complex tasks into smaller sub-tasks. These sub-tasks can then be executed concurrently or sequentially across multiple devices or servers. This approach not only enhances performance but also balances the workload across the network, leveraging the strengths of each device or server involved. Offloading, on the other hand, involves transferring computationally intensive tasks from IoT devices to more powerful edge or cloud servers. This strategy helps alleviate the burden on IoT devices, reduces latency, and capitalizes on the superior computational capacity of centralized servers. The decision to offload tasks depends on dynamic factors such as network conditions, real-time requirements, and energy constraints, ensuring optimal task execution efficiency.

The framework for task partitioning and offloading typically includes three essential components: a profiler, a decision engine, and an offloading agent. The profiler plays a critical role in assessing the hardware conditions of IoT devices, network connectivity status, and energy consumption metrics required for task execution. This information forms the basis for decision-making within the framework. The decision engine utilizes models derived from application dependencies and optimization goals to determine the most effective partitioning and offloading strategy. By modeling task dependencies and evaluating cost models based on performance requirements, the decision engine selects appropriate algorithms for optimal task distribution. Subsequently, the offloading agent facilitates the actual transfer of tasks to remote servers, manages data exchange between devices and servers, and ensures seamless integration of results back into the application. Together, these components form a cohesive framework that enhances the efficiency, responsiveness, and scalability of IoT applications through intelligent task management and resource utilization. Fig. 6 shows the general framework and workflow of task partitioning and offloading.

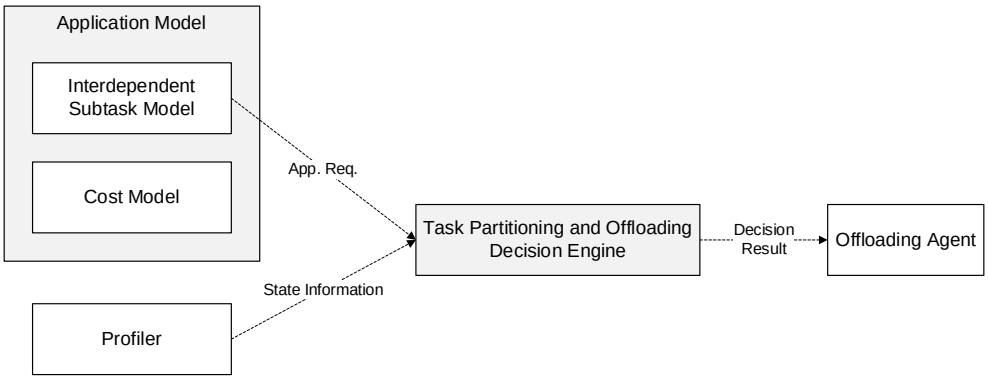

Fig. 6: Framework and workflow of task partitioning and offloading

3.5.2. Data parallelism

Data parallelism focuses on distributing extensive datasets across multiple processing units, enabling simultaneous computations on various data subsets. In the realm of IoT applications, where data streams are often immense and continuous, such as sensor readings and video feeds, data parallelism significantly enhances throughput and scalability. Dividing data processing tasks among numerous computing nodes or devices accelerates analytics and facilitates real-time decision-making. Common techniques for implementing data parallelism in IoT environments include map-reduce frameworks, distributed processing libraries like Apache Spark, and stream processing architectures. These methodologies allow for the concurrent execution of the same operation on different pieces of distributed data, resulting in substantial performance improvements in handling large datasets.

The benefits of data parallelism are manifold. Firstly, it dramatically speeds up computation by processing multiple data points simultaneously, which is crucial for the timely analysis of vast IoT data streams. Secondly, it ensures efficient resource utilization by leveraging the processing capabilities of all available devices, thereby maximizing throughput. Finally, data parallelism offers impressive scalability; as data volumes grow, the parallel processing capacity can be expanded to maintain optimal performance. This scalability is essential in IoT environments where data influx is continuous and increasing, ensuring that processing power can keep pace with data generation rates.

3.5.3. Model parallelism

Model parallelism provides a crucial solution for executing large machine learning (ML) models on resource-constrained IoT devices. Partitioning the model across multiple computing units allows for the parallel computation of different model segments, such as layers in neural networks. This distribution of tasks enables each device or server to handle a smaller portion of the model, thereby reducing the memory load and enhancing the speed of inference tasks. As a result, complex analytics become feasible at the edge of IoT networks, where individual devices might lack the capacity to process entire models independently (Kasarapu et al., 2023). This approach not only makes sophisticated ML applications accessible in resource-limited environments but also optimizes the utilization of available computational resources.

The primary benefits of model parallelism include the ability to handle extremely large models that would otherwise exceed the memory capacity of a single device. By splitting the model into manageable parts, each device only needs to store and process a fraction of the overall model, leading to significant memory efficiency. Additionally, this distribution enhances performance by reducing training times, as multiple devices work concurrently on different parts of the model. This parallel processing capability is particularly valuable for training large-scale neural networks, where the demands for memory and computational power are substantial. Consequently, model parallelism not only enables the development and deployment of advanced ML models in memory-constrained settings but also fosters faster and more efficient training processes.

3.5.4. A novel framework for adaptive parallel processing in IoT (APP-IoT)

In the present work, we have proposed a new framework called Adaptive Parallel Processing for IoT (APP-IoT) (Deb et al., 2022). This framework is developed to address the limitations of traditional static processing techniques. The method enables context-aware, dynamic parallelism to deal with the constraints and capabilities of IoT environments. Unlike conventional models that rigidly apply a single type of parallelism, APP-IoT dynamically selects and adjusts parallel processing strategies, such as task, data, and model parallelism, based on real-time parameters including device memory, CPU load, battery levels, and network latency. The core of this framework is a lightweight context profiler that continuously monitors the status of each device and updates its execution profiles (Li et al., 2023). The profiles are then used by a parallel strategy selector, which determines the most energy-efficient and performance-optimized approach for a given computational task. The flowchart of APP-IoT is shown in Fig. 7. A resource-aware scheduler within APP-IoT maps subtasks to appropriate devices or edge/cloud nodes, factoring in limitations like battery life and communication overhead. Moreover, the framework combines a feedback optimization loop that refines task allocation strategies using performance metrics such as processing time, energy consumption, and latency (Mohammadabadi et al., 2024). This loop introduces an element of adaptive learning, allowing the system to evolve its behavior across multiple execution cycles. As part of its classification, APP-IoT distinguishes between static, reactive, predictive, and collaborative parallelism models. Static parallelism is suited for predictable workloads with fixed topology, while reactive parallelism responds to real-time changes such as energy depletion or bandwidth drops. Predictive parallelism utilizes historical data and machine learning models to forecast optimal strategies, and collaborative parallelism enables multiple devices to jointly execute tasks and share partial results (Kushwaha et al., 2023).

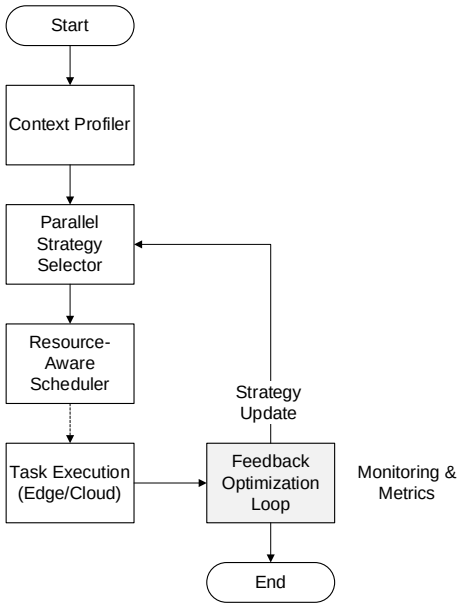

Fig. 7: Adaptive parallel processing in IoT

3.5.5. Energy-efficient parallel processing strategies

Energy efficiency is a critical concern in IoT environments where devices often operate on constrained power sources such as batteries or energy-harvesting modules. The detailed analysis is shown in Table 4. When implementing parallel processing in such devices, traditional methods may inadvertently increase energy consumption due to increased processor usage, memory access, and inter-process communication. Therefore, specialized power-aware strategies are essential to balance performance with energy efficiency. One widely adopted method is dynamic voltage and frequency scaling (DVFS), which adjusts the processor's operating frequency and voltage based on the computational load. The relationship is shown in Fig. 8. Another effective approach is task offloading combined with energy profiling. Further, heterogeneous core utilization in multicore IoT

devices, where tasks are mapped based on energy profiles of individual cores, can minimize energy use. For example, high-load computations are assigned to high-performance cores, while background or less time-sensitive parallel tasks are executed on low-power cores. Memory access also plays a major role in power consumption. Data locality optimization techniques such as loop tiling or memory-aware task partitioning reduce cache misses and DRAM accesses, which are major contributors to static power draw in parallel architectures (Kushwaha et al., 2023; Yang and Luo, 2023).

Table 4: Energy-efficient parallel processing strategies

Strategy	Power saving	Complexity	Suitability
DVFS	High	Low	Sensor hubs
Task offloading	Moderate	Moderate	Edge devices
Energy-aware scheduling	High	High	Gateways
Heterogeneous core usage	Moderate	Moderate	Multicore IoT
Memory optimization	High	Moderate	All devices
Energy-aware compiler	Moderate	High	Advanced systems

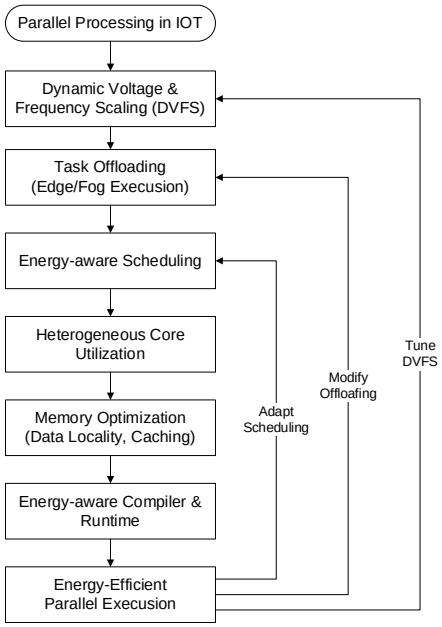

Fig. 8: Energy-efficient strategies

4. Case studies

4.1. Big data processing

The rapid expansion of IoT networks across various domains has underscored the necessity for robust security monitoring systems tailored to the unique requirements of these networks. Traditional security solutions often fall short due to the real-time data analysis and minimal computational overhead demands inherent in IoT environments. Addressing this challenge, Kotenko et al. (2017) proposed a novel architecture leveraging Hadoop and Spark platforms for distributed parallel processing of big data to enhance resource-constrained IoT network security. The architecture encompasses key components for data collection, storage, aggregation, normalization, analysis, and visualization, all of which operate "on-the-fly." Utilizing the Hadoop Distributed File System (HDFS) ensures reliable storage and swift data request processing, crucial for maintaining the system's efficiency within the computational constraints typical of IoT networks.

An extensive experimental evaluation was conducted to assess the performance of the proposed system. Input data streams were synthesized from security events in a segment of an IoT network and an external database of real computer network traffic. Results demonstrated that the Hadoop-based implementation of the system achieved high-performance levels, often surpassing existing solutions.

Furthermore, when deployed on the Spark platform, the system's performance surged approximately tenfold, provided adequate RAM was available. This significant improvement highlights the effectiveness of parallel processing in managing and analyzing vast amounts of IoT-generated data, offering a scalable and efficient solution for enhancing IoT network security. Table 5 displays the findings of the comparative evaluation conducted by Kotenko et al. (2017).

4.2. Malware detection and resource optimization

The widespread integration of IoT devices has significantly enhanced connectivity and computational capabilities, fostering seamless communication across networks. Despite their global deployment, IoT devices are frequently targeted for security breaches due to inherent vulnerabilities. Malware poses a particularly significant risk, exacerbated by the lack of built-in security features and limited resources, which complicate the implementation of effective detection techniques. Traditional methods often assume access to all device resources, an assumption impractical for IoT devices in critical real-world scenarios. Addressing this challenge, a novel approach to malware detection has been introduced by Kasarapu et al. (2024), leveraging resource and workload awareness inspired by model parallelism. Initially, a lightweight regression model assesses available resources for malware detection. Based on resource availability, ongoing workload, and communication costs, the detection task is dynamically allocated either on-device or offloaded to neighboring IoT nodes with sufficient resources. To ensure data integrity and user privacy, the classifier is divided and distributed across multiple nodes, integrating at the parent node for final detection. Experimental results demonstrate a substantial speedup of 9.8x compared to on-device inference while maintaining a high malware detection accuracy of 96.7%.

Table 5: Comparative evaluation of findings

Considered systems	Configuration of the computing platform	Throughput of big data processing (events/sec)	Solved tasks
Vehicle traffic management system (Zygouras et al., 2015)	Hadoop, Storm, and Esper; 3, 5, or 7 VMs in the virtual cluster	7.0×10^4 to 9.0×10^4	Managing the flow of commuting public vehicles
Experimental system SASE++ (Zhang et al., 2014)	Hadoop, optimizer	3.0×10^5 to 7.0×10^6	Ordering sets
Medical information analysis system (Kim and Yu, 2015)	Hadoop	1.2×10^4	Analysis of medical records
System of Kotenko et al. (2017)	Hadoop, Spark; 3, 5, or 7 VMs in the virtual cluster	Hadoop: 1.1×10^5 to 2.1×10^5; Spark: 2.7×10^5 to 1.5×10^6	Network safety for the IoT

Furthermore, the proposed approach combines adaptive model parallelism with resource optimization to enhance the performance of deep learning-based malware detection on IoT devices (Fig. 9).

By converting IoT device firmware into image representations, the researchers utilize deep learning models for malware detection, tailored to the resource constraints of IoT devices. An adaptive model parallelism strategy dynamically partitions the deep learning model across multiple processing units, optimizing the use of available computing resources. Additionally, memory and energy optimization techniques further improve the system's overall performance. Comparative experiments reveal that this innovative approach significantly enhances the accuracy and efficiency of malware detection on IoT devices, outperforming traditional methods that do not leverage these advanced techniques.

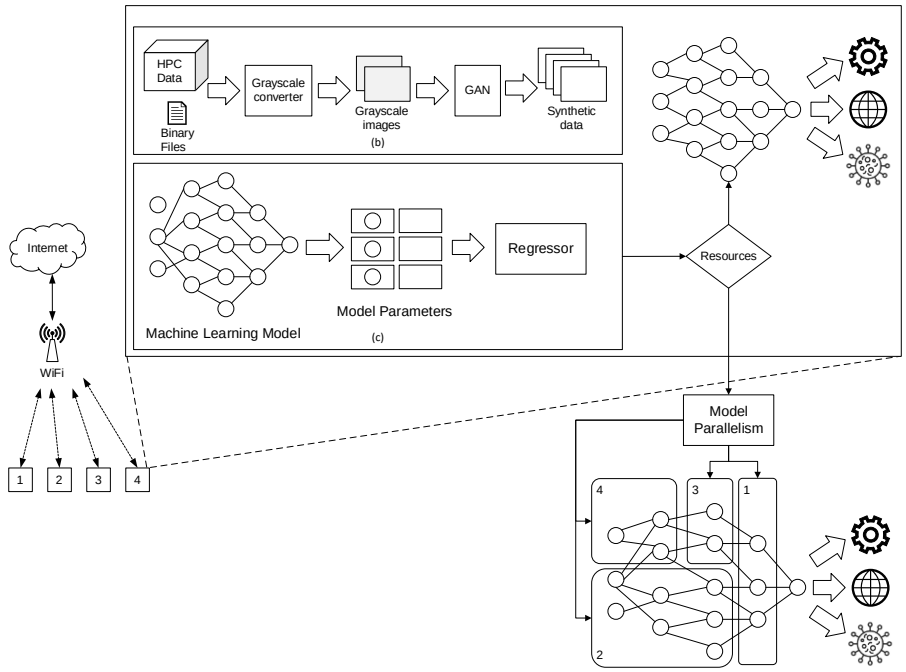

Fig. 9: Model parallelism process (Kasarapu et al., 2024)

5. Challenges and limitations

The parallel processing of resource-constrained IoT devices poses several unique challenges, particularly in terms of scalability. As the number of IoT devices and sensors in a network increases, managing resource constraints becomes a critical issue. IoT devices often have limited processing power, memory, and energy, making it difficult to efficiently scale the network. Moreover, the increase in devices leads to higher data traffic, which can overwhelm network infrastructure, resulting in latency and potential data loss. Efficient data management is also a significant challenge, as handling the vast amount of data generated by numerous devices requires robust data processing and storage solutions. To address scalability, optimizing resource usage, employing edge computing strategies, and designing hierarchical architectures to distribute the processing load are essential (Hong and Varghese, 2019).

Communication overhead is another major challenge in parallel processing for IoT devices. Frequent communication between devices can introduce network latency, especially in low-power and lossy networks typical of IoT environments. Communication is also the most energy-intensive operation for IoT devices, and excessive communication can quickly deplete the limited battery life of these devices (Sathish Kumar et al., 2022; Tao et al., 2016). Additionally, ensuring data consistency and synchronization across multiple devices can be complex and resource-intensive, leading to potential delays and errors. To mitigate communication overhead, it is crucial to optimize communication protocols, reduce the frequency of data exchanges, and employ local processing to minimize the need for inter-device communication.

Programming complexity further complicates parallel processing tasks for resource-constrained IoT devices. The heterogeneity of IoT devices, which vary widely in hardware capabilities, operating systems, and communication protocols, makes developing standardized and interoperable software solutions challenging. Managing concurrent tasks and data flows on limited hardware requires advanced programming techniques and careful resource management. Debugging and testing are also more complex due to the interactions between multiple devices and the potential for non-deterministic behavior. To address these complexities, developers can use middleware platforms, standardized protocols, and development frameworks specifically designed for IoT environments (Razzaque et al., 2015). Additionally, employing simulation tools can help in testing and debugging parallel processing applications before deployment.

6. Future directions

Edge computing offers a promising solution for enhancing parallel processing in IoT resource-constrained devices. By processing data closer to the source, edge computing reduces latency, bandwidth usage, and the need for centralized data processing. This is particularly beneficial for IoT devices, which often operate in environments with limited connectivity and computational power. Integrating edge computing allows IoT devices to perform more complex parallel processing tasks locally, improving real-time decision-making and overall system efficiency (Escamilla-Ambrosio et al., 2018; Ray et al., 2019). Furthermore, edge computing can distribute workloads across multiple devices, optimizing resource utilization and enabling more robust parallel processing capabilities.

Hardware acceleration is a critical avenue for enhancing parallel processing in IoT devices. Utilizing specialized hardware components, such as Graphics Processing Units (GPUs) and Field-Programmable Gate Arrays (FPGAs), can significantly boost the processing power of IoT devices (Molanes et al., 2018). These components are designed to handle parallel tasks more efficiently than general-purpose CPUs, offering substantial performance improvements for compute-intensive applications (Teodoro et al., 2009). Implementing hardware acceleration in IoT devices can enable faster data processing, lower power consumption, and the ability to handle more complex tasks. This approach is particularly advantageous for resource-constrained environments, where maximizing efficiency and performance is crucial.

Energy efficiency is a paramount concern in IoT devices due to their often-limited power sources. Future advancements in parallel processing for IoT devices must prioritize energy efficiency to extend device lifespan and ensure sustainable operation. Techniques such as dynamic voltage and frequency scaling (DVFS), energy-aware scheduling, and low-power design methodologies can be employed to optimize energy consumption during parallel processing tasks (Calore et al., 2017). Additionally, leveraging energy-efficient algorithms and hardware architectures tailored for IoT applications can further reduce power usage. By focusing on energy-efficient parallel processing, IoT devices can achieve higher performance while maintaining low power consumption, making them more viable for long-term deployment in diverse environments.

Future research should also focus on developing lightweight and adaptive parallel processing frameworks specifically designed for embedded IoT hardware. Incorporating edge AI and federated learning can enable privacy-preserving model execution while reducing reliance on centralized infrastructure. Efforts should also target minimizing communication latency through optimized scheduling and protocol design, as well as leveraging hardware accelerators like GPUs, TPUs, and TinyML platforms for energy-aware execution.

7. Conclusions

The exploration of parallel processing techniques specifically tailored for resource-constrained IoT devices reveals significant advancements and notable research gaps. While existing methods have made strides in enhancing computational speed and task distribution, there remains a critical need to optimize energy efficiency without compromising performance. Current literature often focuses on theoretical models or simulations, with insufficient emphasis on practical, scalable solutions for diverse IoT deployments.

To address this gap, a holistic approach integrating energy-aware parallel processing algorithms with adaptive power management strategies is essential. This approach should be tailored to the limited computational and energy resources of IoT devices. Key strategies such as task partitioning and offloading, data parallelism, and model parallelism offer promising avenues for improving performance and efficiency. Task partitioning and offload balance workloads across heterogeneous IoT environments, leveraging the strengths of both edge and cloud servers. Data parallelism enhances throughput and scalability by distributing extensive datasets across multiple processing units, facilitating real-time analytics. Model parallelism allows for the execution of large machine-learning models by partitioning tasks across multiple devices, optimizing memory load and inference speed.

Case studies underscore the practical applications and benefits of these techniques. For instance, employing Hadoop and Spark platforms for distributed parallel processing significantly improves IoT network security, while adaptive model parallelism enhances malware detection efficiency and accuracy on IoT devices. Despite these advancements, challenges such as scalability, communication overhead, and programming

complexity persist. Addressing these issues requires optimizing resource usage, reducing communication frequency, and employing standardized protocols and development frameworks.

Future research directions should focus on integrating edge computing to process data closer to the source, utilizing hardware acceleration for performance boosts, and prioritizing energy-efficient techniques to extend device lifespan. By advancing these areas, we can develop practical, scalable solutions that fully harness the potential of parallel processing for resource-constrained IoT devices, ensuring their viability for long-term deployment across diverse environments.

List of abbreviations

6LoWPAN	IPv6 over low-power wireless personal area networks
APP-IoT	Adaptive parallel processing for IoT
BFS	Breadth-first search
CEP	Complex event processing
CPU	Central processing unit
CoAP	Constrained application protocol
DFS	Depth-first search
DRAM	Dynamic random-access memory
DTLS	Datagram transport layer security
DVFS	Dynamic voltage and frequency scaling
FPGA	Field-programmable gate array
GPU	Graphics processing unit
HDFS	Hadoop distributed file system
IETF	Internet engineering task force
IP	Internet protocol
IPv6	Internet protocol version 6
IoT	Internet of Things
MIMD	Multiple instruction, multiple data
ML	Machine learning
NFC	Near-field communication
PC	Personal computer
RAM	Random-access memory
RFID	Radio-frequency identification
ROM	Read-only memory
SIMD	Single instruction, multiple data
SPMD	Single program, multiple data
SQL	Structured query language
TLS	Transport layer security
TPU	Tensor processing unit
UDP	User datagram protocol
VM	Virtual machine
YARN	Yet another resource negotiator

Acknowledgment

The authors extend their appreciation to the Deanship of Scientific Research at Northern Border University, Arar, KSA, for funding this research work through the project number "NBU-FFR-2025-1260-03."

Compliance with ethical standards

Conflict of interest

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

References

Ahmad T and Zhang D (2021). Using the Internet of Things in smart energy systems and networks. Sustainable Cities and Society, 68: 102783. https://doi.org/10.1016/j.scs.2021.102783

Al-Kashoash HA, Kharrufa H, Al-Nidawi Y, and Kemp AH (2019). Congestion control in wireless sensor and 6LoWPAN networks: Toward the Internet of Things. Wireless Networks, 25(8): 4493-4522. https://doi.org/10.1007/s11276-018-1743-y

Allioui H and Mourdi Y (2023). Exploring the full potentials of IoT for better financial growth and stability: A comprehensive survey. Sensors, 23(19): 8015. https://doi.org/10.3390/s23198015 **PMid:37836845 PMCid:PMC10574902**

Amadeo M, Campolo C, Quevedo J, Corujo D, Molinaro A, Iera A, Aguiar RL, and Vasilakos AV (2016). Information-centric networking for the Internet of Things: Challenges and opportunities. IEEE Network, 30(2): 92-100. https://doi.org/10.1109/MNET.2016.7437030

Amrahov SE, Ar Y, Tugrul B, Akay BE, and Kartli N (2024). A new approach to Mergesort algorithm: Divide smart and conquer. Future Generation Computer Systems, 157: 330-343. https://doi.org/10.1016/j.future.2024.03.049

Ashton K (2009). That 'Internet of Things' thing. RFID Journal, 22(7): 97-114.

Atzori L, Iera A, and Morabito G (2017). Understanding the internet of things: definition, potentials, and societal role of a fast evolving paradigm. Ad Hoc Networks, 56: 122-140. https://doi.org/10.1016/j.adhoc.2016.12.004

Bibri SE (2018). The IoT for smart sustainable cities of the future: An analytical framework for sensor-based big data applications for environmental sustainability. Sustainable Cities and Society, 38: 230-253. https://doi.org/10.1007/978-3-319-73981-6

Calore E, Gabbana A, Schifano SF, and Tripiccione R (2017). Evaluation of DVFS techniques on modern HPC processors and accelerators for energy-aware applications. Concurrency and Computation: Practice and Experience, 29(12): e4143. https://doi.org/10.1002/cpe.4143

Chander B and Kumaravelan G (2019). Internet of Things: Foundation. In: Peng SL, Pal S, and Huang L (Eds.), Principles of internet of things (IoT) ecosystem: Insight paradigm: 3-33. Springer International Publishing, Cham, Switzerland. https://doi.org/10.1007/978-3-030-33596-0_1

Deb PK, Mukherjee A, Singh D, and Misra S (2022). Loop-the-loops: Fragmented learning over networks for constrained IoT devices. IEEE Transactions on Parallel and Distributed Systems, 34(1): 316-327. https://doi.org/10.1109/TPDS.2022.3220221

Deep S, Zheng X, Jolfaei A, Yu D, Ostovari P, and Kashif Bashir A (2022). A survey of security and privacy issues in the Internet of Things from the layered context. Transactions on Emerging Telecommunications Technologies, 33(6): e3935. https://doi.org/10.1002/ett.3935

Devasena CL (2016). IPv6 low power wireless personal area network (6LoWPAN) for networking Internet of Things (IoT)–Analyzing its suitability for IoT. Indian Journal of Science and Technology, 9(30): 1-6. https://doi.org/10.17485/ijst/2016/v9i30/98730

Escamilla-Ambrosio PJ, Rodríguez-Mota A, Aguirre-Anaya E, Acosta-Bermejo R, and Salinas-Rosales M (2018). Distributing computing in the Internet of Things: Cloud, fog and edge computing overview. In the NEO 2016: Results of the Numerical and Evolutionary Optimization Workshop NEO 2016 and the NEO Cities 2016 Workshop, Springer International Publishing, Tlalnepantla, Mexico: 87-115. https://doi.org/10.1007/978-3-319-64063-1_4

Farhan MN, Habib MA, and Ali MA (2018). A study and performance comparison of MapReduce and Apache Spark on Twitter data on Hadoop cluster. International Journal of Information Technology and Computer Science (IJITCS), 10(7): 61-70. https://doi.org/10.5815/ijitcs.2018.07.07

Fossati T and Tschofenig H (2016). Transport layer security (TLS)/datagram transport layer security (DTLS) profiles for the Internet of Things. IETF RFC 7925. https://doi.org/10.17487/RFC7925

Fuentes-Samaniego RA, La VH, Cavalli AR, Nolazco-Flores JA, and Ramirez-Velarde RV (2019). A monitoring-based approach for WSN security using IEEE-802.15. 4/6LowPAN and DTLS communication. International Journal of Autonomous and Adaptive Communications Systems, 12(3): 218-243. https://doi.org/10.1504/IJAACS.2019.10022471

Hong CH and Varghese B (2019). Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms. ACM Computing Surveys (CSUR), 52(5): 1-37. https://doi.org/10.1145/3326066

Ijemaru GK, Ang KLM, and Seng JK (2022). Wireless power transfer and energy harvesting in distributed sensor networks: Survey, opportunities, and challenges. International Journal of Distributed Sensor Networks, 18(3). https://doi.org/10.1177/15501477211067740

Jeyaraj R, Balasubramaniam A, MA AK, Guizani N, and Paul A (2023). Resource management in cloud and cloud-influenced technologies for Internet of Things applications. ACM Computing Surveys, 55(12): 1-37. https://doi.org/10.1145/3571729

Kahira AN, Nguyen TT, Gomez LB, Takano R, Badia RM, and Wahib M (2021). An oracle for guiding large-scale model/hybrid parallel training of convolutional neural networks. In the Proceedings of the 30th International Symposium on High-Performance Parallel and Distributed Computing, ACM, Virtual Event, Sweden: 161-173. https://doi.org/10.1145/3431379.3460644

Kan C, Yang H, and Kumara S (2018). Parallel computing and network analytics for fast industrial Internet-of-Things (IIoT) machine information processing and condition monitoring. Journal of Manufacturing Systems, 46: 282-293. https://doi.org/10.1016/j.jmsy.2018.01.010

Kasarapu S, Shukla S, and Dinakarrao SMP (2024). Enhancing IoT malware detection through adaptive model parallelism and resource optimization. Arxiv Preprint Arxiv:2404.08808. https://doi.org/10.48550/arXiv.2404.08808

Kennedy RK, Khoshgoftaar TM, Villanustre F, and Humphrey T (2019). A parallel and distributed stochastic gradient descent implementation using commodity clusters. Journal of Big Data, 6: 16. https://doi.org/10.1186/s40537-019-0179-2

Ketu S, Mishra PK, and Agarwal S (2020). Performance analysis of distributed computing frameworks for big data analytics: Hadoop vs Spark. Computación y Sistemas, 24(2): 669-686. https://doi.org/10.13053/cys-24-2-3401

Khalil K, Elgazzar K, Seliem M, and Bayoumi M (2020). Resource discovery techniques in the Internet of Things: A review. Internet of Things, 12: 100293. https://doi.org/10.1016/j.iot.2020.100293

Khattak SBA, Nasralla MM, Farman H, and Choudhury N (2023). Performance evaluation of an IEEE 802.15. 4-based thread network for efficient Internet of Things communications in smart cities. Applied Sciences, 13(13): 7745. https://doi.org/10.3390/app13137745

Kim MJ and Yu YS (2015). Development of real-time big data analysis system and a case study on the application of information in a medical institution. International Journal of Software Engineering and Its Applications, 9(7): 93-102. https://doi.org/10.14257/ijseia.2015.9.7.10

Kotenko IV, Saenko I, and Kushnerevich A (2017). Parallel big data processing system for security monitoring in Internet of Things networks. Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, 8(4): 60-74. https://doi.org/10.15622/sp.59.1

Krishnamoorthy S, Dua A, and Gupta S (2023). Role of emerging technologies in future IoT-driven Healthcare 4.0 technologies: A survey, current challenges and future directions. Journal of Ambient Intelligence and Humanized Computing, 14(1): 361-407. https://doi.org/10.1007/s12652-021-03302-w

Kushwaha D, Redhu S, Brinton CG, and Hegde RM (2023). Optimal device selection in federated learning for resource-constrained edge networks. IEEE Internet of Things Journal, 10(12): 10845-10856. https://doi.org/10.1109/JIOT.2023.3243082

Lee I and Lee K (2015). The Internet of Things (IoT): Applications, investments, and challenges for enterprises. Business Horizons, 58(4): 431-440. https://doi.org/10.1016/j.bushor.2015.03.008

Li Y, Ge X, Lei B, Zhang X, and Wang W (2023). Joint task partitioning and parallel scheduling in device-assisted mobile edge networks. IEEE Internet of Things Journal, 11(8): 14058-14075. https://doi.org/10.1109/JIOT.2023.3341062

Liu H and Huang HH (2015). Enterprise: Breadth-first graph traversal on GPUs. In the Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, ACM, Austin, USA: 1-12. https://doi.org/10.1145/2807591.2807594

Liyanage NH (2017). Comprehensive comparison of parallel sorting techniques, architectures and behaviors which support for distributed environments. In the International Conference on Big Data Analytics and Computational Intelligence, IEEE, Chirala, Andhra Pradesh, India: 412-417. https://doi.org/10.1109/ICBDACI.2017.8070874

Makhdoom I, Abolhasan M, Abbas H, and Ni W (2019). Blockchain's adoption in IoT: The challenges, and a way forward. Journal of Network and Computer Applications, 125: 251-279. https://doi.org/10.1016/j.jnca.2018.10.019

Malik PK, Sharma R, Singh R, Gehlot A, Satapathy SC, Alnumay WS, Pelusi D, Ghosh U, and Nayak J (2021). Industrial internet of things and its applications in Industry 4.0: State of the art. Computer Communications, 166: 125-139. https://doi.org/10.1016/j.comcom.2020.11.016

Mohammadabadi SMS, Zawad S, Yan F, and Yang L (2024). Speed up federated learning in heterogeneous environments: A dynamic tiering approach. IEEE Internet of Things Journal, 12(5): 5026–5035. https://doi.org/10.1109/JIOT.2024.3487473

Molanes RF, Amarasinghe K, Rodriguez-Andina J, and Manic M (2018). Deep learning and reconfigurable platforms in the Internet of Things: Challenges and opportunities in algorithms and hardware. IEEE Industrial Electronics Magazine, 12(2): 36-49. https://doi.org/10.1109/MIE.2018.2824843

Mourtzis D, Vlachou E, Milas N, and Xanthopoulos N (2016). A cloud-based approach for maintenance of machine tools and equipment based on shop-floor monitoring. Procedia CIRP, 41: 655-660. https://doi.org/10.1016/j.procir.2015.12.069

Nižetić S, Šolić P, Gonzalez-De DLDI, and Patrono L (2020). Internet of Things (IoT): Opportunities, issues and challenges towards a smart and sustainable future. Journal of Cleaner Production, 274: 122877. https://doi.org/10.1016/j.jclepro.2020.122877 **PMid:32834567 PMCid:PMC7368922**

Raghavan NS and Waghmare T (2002). DPAC: An object-oriented distributed and parallel computing framework for manufacturing applications. IEEE Transactions on Robotics and Automation, 18(4): 431-443. https://doi.org/10.1109/TRA.2002.802236

Raj M, Gupta S, Chamola V, Elhence A, Garg T, Atiquzzaman M, and Niyato D (2021). A survey on the role of Internet of Things for adopting and promoting Agriculture 4.0. Journal of Network

and Computer Applications, 187: 103107. https://doi.org/10.1016/j.jnca.2021.103107

Ray PP, Dash D, and De D (2019). Edge computing for Internet of Things: A survey, e-healthcare case study and future direction. Journal of Network and Computer Applications, 140: 1-22. https://doi.org/10.1016/j.jnca.2019.05.005

Razzaque MA, Milojevic-Jevric M, Palade A, and Clarke S (2015). Middleware for Internet of Things: A survey. IEEE Internet of Things Journal, 3(1): 70-95. https://doi.org/10.1109/JIOT.2015.2498900

Sanislav T, Mois GD, Zeadally S, and Folea SC (2021). Energy harvesting techniques for Internet of Things (IoT). IEEE Access, 9: 39530-39549. https://doi.org/10.1109/ACCESS.2021.3064066

Sathish Kumar L, Ahmad S, Routray S, Prabu AV, Alharbi A, Alouffi B, and Rajasoundaran S (2022). Modern energy optimization approach for efficient data communication in IoT-based wireless sensor networks. Wireless Communications and Mobile Computing, 2022: 7901587. https://doi.org/10.1155/2022/7901587

Sehgal A, Perelman V, Kuryla S, and Schonwalder J (2012). Management of resource constrained devices in the Internet of Things. IEEE Communications Magazine, 50(12): 144-149. https://doi.org/10.1109/MCOM.2012.6384464

Shuvo MMH, Islam SK, Cheng J, and Morshed BI (2022). Efficient acceleration of deep learning inference on resource-constrained edge devices: A review. Proceedings of the IEEE, 111(1): 42-91. https://doi.org/10.1109/JPROC.2022.3226481

Tao F, LaiLi Y, Xu L, and Zhang L (2012). FC-PACO-RM: A parallel method for service composition optimal-selection in cloud manufacturing system. IEEE Transactions on Industrial Informatics, 9(4): 2023-2033. https://doi.org/10.1109/TII.2012.2232936

Tao F, Wang Y, Zuo Y, Yang H, and Zhang M (2016). Internet of Things in product life-cycle energy management. Journal of Industrial Information Integration, 1: 26-39. https://doi.org/10.1016/j.jii.2016.03.001

Teodoro G, Sachetto R, Sertel O, Gurcan MN, Meira W, Catalyurek U, and Ferreira R (2009). Coordinating the use of GPU and CPU for improving performance of compute intensive applications. In the IEEE International Conference on Cluster Computing and Workshops, IEEE. New Orleans, USA: 1-10. https://doi.org/10.1109/CLUSTR.2009.5289193

Yang D and Luo Z (2023). A parallel processing CNN accelerator on embedded devices based on optimized MobileNet. IEEE Internet of Things Journal, 10(21): 18844-18852. https://doi.org/10.1109/JIOT.2023.3277869

Yang HC, Dasdan A, Hsiao RL, and Parker DS (2007). Map-reduce-merge: simplified relational data processing on large clusters. In the Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, ACM, Beijing, China: 1029-1040. https://doi.org/10.1145/1247480.1247602

Yuan MM, Baker JW, and Meilander WC (2013). Comparisons of air traffic control implementations on an associative processor with a MIMD and consequences for parallel computing. Journal of Parallel and Distributed Computing, 73(2): 256-272. https://doi.org/10.1016/j.jpdc.2012.05.009

Zhang H, Diao Y, and Immerman N (2014). On complexity and optimization of expensive queries in complex event processing. In the SIGMOD '14: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, Snowbird, USA: 217-228. https://doi.org/10.1145/2588555.2593671

Zhang Y, Tsai PA, and Tseng HW (2022). SIMD2: A generalized matrix instruction set for accelerating tensor computation beyond GEMM. In the Proceedings of the 49th Annual International Symposium on Computer Architecture, ACM, New York, USA: 552-566. https://doi.org/10.1145/3470496.3527411

Ziegler S, Kirstein P, Ladid L, Skarmeta A, and Jara A (2015). The case for IPv6 as an enabler of the Internet of Things. IEEE Internet of Things. Available online at: http://iot.ieee.org/newsletter/july-2015/the-case-for-ipv6-as-an-enabler-of-the-internet-of-things.html

Zygouras N, Zacheilas N, Kalogeraki V, Kinane D, and Gunopulos D (2015). Insights on a scalable and dynamic traffic management system. In the 18th International Conference on Extending Database Technology (EDBT), Brussels, Belgium: 653-664. https://doi.org/10.5441/002/edbt.2015.65