IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 25 September 2025, accepted 25 October 2025, date of publication 30 October 2025, date of current version 6 November 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3627127

SURVEY

Bits to Qubits: A Comparative Study of Memory Management in Classical and Quantum Systems

PRADEEP LAMICHHANE AND DANDA B. RAWAT, (Senior Member, IEEE)
Department of Electrical Engineering and Computer Science, Howard University, Washington, DC 20059, USA

Corresponding author: Danda B. Rawat (danda.rawat@howard.edu)

This work was supported in part by the Department of Defense (DoD)/Department of War (DoW) Center of Excellence in Artificial Intelligence and Machine Learning (CoE-AIML) at Howard University with U.S. Army Research Laboratory under Contract W911NF-20-2-0277 as well as Meta, VMware and Microsoft Corporation Research Gift Funds.

ABSTRACT In the near future, as quantum and classical systems coexist, managing hybrid systems will require efficient conversion between bits and qubits. Therefore, understanding memory management for both classical and quantum systems is essential. Along that line, this survey provides a comparative analysis of memory management in classical and quantum architectures. The traditional memory hierarchy and allocation strategies used in classical systems are examined along with caching and paging techniques which optimize performance and resource utilization. On the other hand, quantum memory management focuses on addressing unique challenges such as quantum decoherence and error correction using advanced techniques such as Shor's code, Surface codes, and other specialized error correction codes. This study explores Quantum Random Access Memory (QRAM) and its role in enabling efficient data retrieval in quantum systems. Quantum read/write operations and their impact on coherence and fidelity are analyzed, highlighting advancements in qubit state measurement and error mitigation. While classical systems take benefits of deterministic and hierarchical memory structures, quantum systems must overcome environmental noise and decoherence to maintain data stability. Furthermore, scalability issues and emerging trends in quantum architecture are discussed, emphasizing how advancements in Quantum Error Correction (QEC) and hybrid systems can address current limitations in memory management.

INDEX TERMS Memory management, classical computing, quantum computing, qubits, quantum random access memory, quantum error correction, quantum read/write operations, memory hierarchy, hybrid quantum-classical systems, computational performance.

I. INTRODUCTION

Memory management is important in both classical and quantum computing, but each domain handles it in very different ways. As quantum technologies advance and hybrid systems emerge, it becomes important to compare how memory is managed in both types of systems. Memory management has a crucial role in computational performance, as it directly impacts the speed, efficiency, and reliability of the system. Efficient memory management ensures optimal resource utilization, prevents latency, and enables faster data access, which is essential for modern data-intensive applications. These fundamental principles apply to both classical and quantum systems, where different memory management techniques address specific performance and energy efficiency requirements. Effective memory allocation, caching, and garbage collection techniques in classical computing optimize resource usage and reduce latency, which is essential for handling large-scale data and high-performance applications [1], [2]. Advanced memory management, such as dynamic memory allocation, minimizes fragmentation and improves computational stability [3].

Building on these established techniques, modern advancements have further enhanced memory management for high-demand applications. Applying self-awareness principles enabled adaptive and energy-efficient memory systems, achieving 10.5% energy savings in on-chip L1 cache [4]. Efficient flash memory management is essential in virtual

The associate editor coordinating the review of this manuscript and approving it for publication was Mario Donato Marino.

memory systems to address energy consumption challenges. Techniques like subpaging, HotCache, and duplication-aware garbage collection can collectively reduce energy usage by 42.2% [5]. These energy-efficient solutions highlight the potential of efficient memory management to solve computational challenges.

Recent advances in GPU memory management have enabled significant performance improvements and optimized resource utilization, particularly for deep learning models. In [6], authors proposed SmartPool and AutoSwap to optimize GPU memory for deep learning by reducing memory fragmentation and swapping unused variables, achieving up to 34.2% memory savings without affecting model accuracy or training speed. SuperNeurons dynamically allocates memory for convolution workspaces and reduces peak memory usage through Liveness Analysis, Unified Tensor Pool, and Cost-Aware Recomputation. SuperNeurons achieved the leading performance among state-of-the-art deep learning systems on GPUs [7]. Capuchin optimizes GPU memory for deep neural network training by dynamically tracking tensor access patterns and enabling fine-grained eviction, prefetching, and recomputation. It reduced memory usage by up to 85%, supported larger batch sizes, and achieved up to 286% speedup compared to Virtualized Deep Neural Networks (vDNN) [8]. A memory management framework for integrated CPU/GPU systems optimized memory usage and performance in autonomous platforms through dynamic policy selection and kernel overlapping [9].

In cloud computing environments, efficient memory management is equally important. In [10], authors explored resource overcommitment in cloud computing and proposed a Distributed Resource Scheduler (DRS). They used memory ballooning and live migration to optimize resource utilization and reduce costs in virtualized environments. In another study, a machine learning-based approach was explored for dynamic resource allocation and memory management in cloud and mobile environments [11]. By eliminating unnecessary cache data, authors were able to improve memory efficiency, reduce overhead, and enhance device performance. This demonstrated scalability and effectiveness in optimizing large-scale systems. In [12], authors emphasized the role of memory management to enhance performance, optimize resource utilization, ensure energy efficiency, and solve the challenges of heterogeneous architectures and diverse workloads in High-Performance Computing (HPC) and Cloud Computing environments.

While classical systems have achieved remarkable efficiency improvements with advanced memory management techniques, the rise of quantum computing introduces an entirely new set of challenges and opportunities for memory management. Unlike classical bits, qubits are unstable due to decoherence and noise issues. Quantum Error Correction (QEC) codes such as Shor's code and Surface codes, are essential for maintaining qubit states and ensuring computational accuracy [13], [14], [15]. The shift in paradigm requires advanced memory strategies to handle the probabilistic nature of quantum systems.

To address these challenges, researchers have proposed innovative quantum memory management techniques that optimize performance and resource utilization in quantum computing. A SAT-based reversible pebbling game was used in [16], which optimized quantum memory management by reducing qubit usage by 52.77% on average while balancing trade-offs between memory and operations. In [17], authors introduced throughput-optimal algorithms for managing quantum switches with memory constraints, maximizing entanglement rates through efficient memory allocation and dynamic scheduling policies. In [18], quantum memory was explored as a key component of future quantum computing, emphasizing its role in addressing scalability challenges through the separation of computing and memory units. The authors presented a design stack for quantum memory and explored its applications as well. Quantum Secure Direct Communication (QSDC) using atomic quantum memory was demonstrated in [19], highlighting its potential for secure, long-distance quantum networks. The importance of memory management in optimizing learning efficiency and adaptability through dynamic memory in quantum reinforcement learning has been highlighted in recent research [20]. These advances demonstrate how quantum memory management is evolving to meet the demands of next-generation computing architectures.

Techniques such as dynamic voltage scaling in classical systems reduce power consumption, while error correction in quantum systems supports fault-tolerant operations [21], [22]. From optimizing classical systems for energy efficiency and scalability to addressing unique challenges in quantum computing, advancements in memory management techniques drive innovation across diverse computing paradigms. Efficient memory management is crucial in hybrid classical-quantum systems to enable seamless integration between deterministic classical computing and probabilistic quantum computing. By optimizing data transfer and memory allocation strategies, these systems can improve scalability, performance, and energy efficiency in high-demand computational environments.

Given the differences in how classical and quantum systems manage memory, a structured comparison is necessary to understand their individual challenges, areas of overlap, and how they can be integrated in hybrid computing environments. This survey addresses this need by providing a comprehensive analysis of memory management in both classical and quantum architectures.

The remainder of this paper is structured as follows. Section II covers overview of classical and quantum systems and scope of our work. Section III reviews memory management in classical computing, including architecture, allocation strategies, and error correction. Section IV covers quantum memory organization, read/write operations, and Quantum Error Correction techniques. Section V presents

a comparative analysis of classical and quantum memory systems across key performance metrics and highlights their interdependencies. Section VI discusses hybrid quantum-classical systems, emerging trends, and future research directions. Finally, Section VII concludes the paper.

II. BACKGROUND AND SCOPE

A. OVERVIEW OF CLASSICAL VS. QUANTUM PARADIGMS

In classical computing, data are represented in bits that can be either 0 or 1 at one instance of time. These systems process information deterministically, following a specific sequence of instructions where each operation has a predictable outcome. On the other hand, quantum computing uses qubits as fundamental units of information, which can exist in a superposition of both 0 and 1 states simultaneously. Mathematically, a qubit is represented as [23]:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$
$$\text{where } \alpha, \beta \in \mathbb{C} \text{ and } |\alpha|^2 + |\beta|^2 = 1. \tag{1}$$

α and β are complex probability amplitudes, and the absolute squares of the amplitudes represent the probabilities of measuring the qubit in states $|0\rangle$ or $|1\rangle$. This probabilistic representation ensures that quantum transformations preserve probability through unitary operations, which are fundamental to quantum computation.

A single-qubit state can also be expressed using the Bloch sphere representation [24], which provides a geometric visualization of all pure qubit states:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle, \tag{2}$$

where θ and ϕ are spherical coordinates. Any pure qubit state corresponds to a point on the surface of the unit sphere, which offers an intuitive picture of quantum state manipulation.

Additionally, quantum entanglement creates non-classical correlations between qubits, such that the measurement outcome of one qubit is directly linked to the state of another. A well-known example is the maximally entangled state [25]:

$$|\psi\rangle = \frac{1}{\sqrt{2}}\left(|x\rangle_1|y\rangle_2 - |y\rangle_1|x\rangle_2\right), \tag{3}$$

where $|x\rangle$ and $|y\rangle$ are basis states. This state illustrates how entanglement prevents a product-state decomposition and is central to many quantum information protocols.

Classical memory systems are well structured, scalable, and energy efficient, leveraging hierarchical architectures to optimize performance for traditional workloads. However, quantum memory systems face significant challenges due to noise, decoherence, and qubit susceptibility to errors. To ensure data integrity, sophisticated error correction techniques are required. While classical systems focus on efficiency and power optimization, quantum systems prioritize minimizing decoherence and operational errors to maintain coherence. Classical memory is designed for general-purpose computing, whereas quantum memory is optimized for specialized applications such as cryptography, optimization, and large-scale simulations, where quantum parallelism offers computational advantages.

B. OBJECTIVES AND SCOPE

Despite fundamental differences in scale, cost, and maturity, comparing classical and quantum memory is essential as hybrid systems become more prevalent. Classical architectures manage terabytes of memory with refined caching and allocation techniques, while quantum systems, constrained to tens or hundreds of qubits, face challenges like decoherence and error correction. These differences highlight the need for a focused comparison to evaluate which classical memory management techniques can be adapted to quantum systems, which ones are fundamentally incompatible, and how coordinated strategies can support efficient memory management in emerging hybrid quantum-classical architectures.

The objective of this work is to provide a comprehensive comparative analysis of memory management in classical and quantum computing architectures. We compare classical and quantum memory metrics to highlight the gaps, analyze hybrid designs that let the two systems work together, and discuss the key thresholds that must be achieved before fully fault-tolerant quantum memory becomes practical.
We structure this analysis around the following research questions (RQs):

- RQ1: What are the core memory management strategies in classical computing, and how do they ensure performance and scalability?
- RQ2: How does quantum memory management differ from classical approaches, particularly in error correction and data stability?
- RQ3: How do quantum read/write operations influence memory coherence and error rates on different qubit platforms?
- RQ4: What fundamental differences exist between classical and quantum memory management when evaluated in terms of latency, energy efficiency, and scalability?
- RQ5: How do hybrid quantum-classical workflows illustrate memory coordination patterns, and what lessons can be drawn for future hybrid architectures?
- RQ6: What are the key challenges and promising directions for research in scalable and energy-efficient quantum memory systems?

The scope of this survey is delimited by the following contributions:

- We examine memory management techniques in classical systems, including memory hierarchy, virtual memory, caching, dynamic allocation, and advanced error correction codes.
- We analyze quantum memory management by covering quantum register, qubit technologies (superconducting, neutral-atom, trapped-ion, photonic, spin), Quantum Random Access Memory (QRAM) models, and

advanced Quantum Error Correction techniques along with Shor's, Steane's, and Surface codes.
- We examine quantum read/write operations and analyze their impact on qubit fidelity and memory coherence across different platforms such as superconducting and silicon-based qubits.
- We compare classical and quantum memory systems by highlighting key differences in latency, energy efficiency, and scalability.
- We discuss hybrid quantum-classical workflows to illustrate memory coordination patterns and the interdependencies of classical pre/post-processing with quantum execution.
- We identify key challenges and outline future research directions in scalable and energy-efficient quantum memory design.

III. CLASSICAL MEMORY MANAGEMENT

Techniques such as hierarchical memory structures, allocation strategies, and caching mechanisms in classical systems are optimized to support high-speed computation and efficient resource utilization across diverse applications [1].

A. MEMORY ARCHITECTURE

Classical memory architectures are structured in hierarchical levels to balance speed and cost. The memory hierarchy shown in Figure 1 corresponds to the classical Von Neumann architecture, where a single memory space stores both data and instructions. While other architectures exist, the Von Neumann model remains the most widely adopted and is used here as the reference point for comparison with quantum systems. In this hierarchy, registers, caches (L1–L3), and main memory (DRAM) form the core memory directly accessed by the processor. Beyond this, secondary storage such as disks (SSD/HDD) and tertiary storage such as tape represent storage and are included to illustrate the broader latency–capacity tradeoff. Data move from slower, high capacity storage, such as disks or tape, to faster, lower capacity memory as needed for processing. This hierarchical structure has an important role in reducing latency and managing data access speeds, enabling efficient processing across different workloads. Authors in [26] presented a comprehensive survey of memory architecture design for 3D Chip Multi-Processors (3D CMPs), emphasizing how 3D integration technologies – such as through-silicon vias (TSVs) – can overcome traditional memory bandwidth limitations known as the "Memory Wall". They categorized prior work into two key areas: stacking cache-only architectures and stacking main memory architectures, highlighting design trade-offs in performance, power, and thermal constraints. Authors also explored the integration of emerging non-volatile memories (MRAM, PCRAM, RRAM), hybrid cache architectures, and reconfigurable memory hierarchies. Additionally, they discuss challenges such as thermal management, TSV overhead, and the need for new memory microarchitectures and coherence protocols in heterogeneous and embedded 3D systems. Authors in [27] proposed a 3D-stacked Dynamic Random Access Memory (DRAM)-based chip multiprocessor architecture incorporating hybrid electrical-optical interconnects and self-adaptive runtime page mapping. Their evaluation demonstrated that this design improved instructions per cycle (IPC) and reduced communication costs, thereby enabling higher core counts with smaller per-core caches while preserving overall performance.

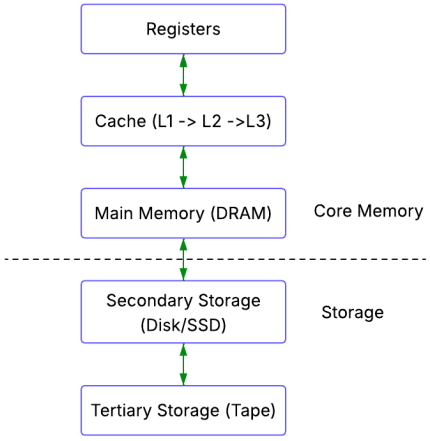

FIGURE 1. Classical Memory Hierarchy. Levels above the dashed line represent core memory directly accessed by the processor(registers, caches, main memory). Levels below represent storage (disk, tape).

1) VIRTUAL MEMORY

Virtual memory adds an abstraction layer to the hierarchical memory system, which allows applications to use more memory than what is physically available. By mapping logical addresses to physical memory, virtual memory enables multitasking, memory protection, and isolation between processes. This abstraction is achieved through techniques like paging and segmentation, which divide memory into manageable units. Paging splits memory into fixed-size blocks called pages, while segmentation organizes memory into variable-sized blocks based on logical divisions such as code, data, and stack [28]. Recent advancements, such as the Distributed Page Table (DPT), further enhance virtual memory systems by addressing the limitations of traditional page table structures. DPT eliminates the need for large contiguous physical memory by distributing page table entries (PTEs) across the entire memory space. This resolves hash collisions through techniques such as Strided Open Addressing and Collision-Aware Virtual Address Allocation, thus improving performance and scalability [29].

2) CACHING AND CACHE OPTIMIZATION

Caching is a critical performance enhancement technique in which the frequently accessed data is stored in smaller memory to reduce access times. Multilevel caches (L1, L2, and L3) allow quick data access by minimizing latency. Machine learning techniques have been applied to enhance cache prediction and management. For example, a learning-

based caching mechanism for edge content delivery used machine learning to forecast content popularity by optimizing cache replacement policies and improving resource utilization [30]. In distributed computing environments, edge caching strategies have been developed to optimize data placement and retrieval. A deep learning-based predictive caching approach, using recurrent neural networks with Long Short Term Memory (LSTM), predicts content demand in edge networks which allows low-latency content delivery to end-users [31].

3) CACHE COHERENCE AND MULTICORE OPTIMIZATION

Cache coherence protocols are important in multiprocessor and multicore systems, ensuring data consistency across caches while enhancing efficiency and reliability [2]. For example, the MESI (Modified, Exclusive, Shared, Invalid) and MOESI (Modified, Owner, Exclusive, Shared, Invalid) protocols ensure data consistency in multiprocessor systems by propagating modifications across caches. In multicore processors, advanced techniques like Non-Uniform Cache Access (NUCA) optimize data placement by dynamically migrating frequently accessed data closer to the relevant core, reducing latency. Furthermore, hybrid cache memory systems that combine volatile and non-volatile memory such as SRAM and MRAM, address challenges such as energy efficiency and consistency in multicore architectures [32]. These innovations contribute to scalable and high-performance systems while maintaining cache coherence across complex cache hierarchies.

B. MEMORY ACCESS AND ALLOCATION

Memory allocation is the process of reserving a part of a computer's memory for use by programs or applications. In classical computing, memory allocation is broadly divided into Static Allocation and Dynamic Allocation. Static memory allocation takes place at compile time, where the memory size of variables, programs, or applications is fixed before execution begins. While static allocation offers faster execution and avoids fragmentation issues, it lacks flexibility and can lead to inefficient memory usage when pre-allocated space is underutilized. On the other hand, dynamic memory allocation takes place at run-time, where memory is allocated based on real-time demands, allowing efficient use of available memory resources. The flexibility makes dynamic allocation better applicable for scenarios that involve unpredictable workloads and growing data requirements [33]. However, dynamic allocation introduces challenges such as memory leaks, dangling pointers, and external fragmentation, which require proper management strategies to ensure optimal performance.

1) MEMORY POOLING

Memory pooling is a technique that disaggregates memory resources from individual systems, which allows them to be shared and accessed dynamically across multiple hosts. It enhances memory utilization, elasticity, and resource efficiency, addressing challenges such as fragmentation and over-provisioning. In [34], authors used Compute Express Link (CXL) for memory pooling, introducing DirectCXL to enable efficient and low latency access to disaggregated memory resources. Similarly, STARNUMA was introduced in [35], which leveraged a CXL-attached memory pool to mitigate Non-Uniform Memory Access (NUMA) challenges by placing heavily shared ''vagabond pages'' in the pool. This approach reduced the average memory access time by 48% and improved performance by up to 2.17× in large multi-socket systems. In [36], authors presented a fast fixed-size memory pool manager optimized for time-critical systems. They eliminated loops and minimized overhead while achieving $O(1)$ memory allocation and deallocation with reduced fragmentation.

2) GARBAGE COLLECTION

Garbage collection (GC) is a process that prevents memory leaks by reclaiming memory that is no longer in use. Techniques like mark-and-sweep, generational GC, and reference counting are common approaches for garbage collection in classical systems. These methods help maintain system performance by ensuring that memory is reused efficiently and that unused data do not consume valuable resources [3].

Efficient memory management is essential to maintain computational performance, reduce power consumption, and support multitasking in classical systems. These techniques, which are developed over decades, form a robust foundation for handling the predictable and high-speed demands of the classical computing environment.

C. ERROR DETECTION AND CORRECTION

Error detection and correction have an important role in ensuring the reliability of classical memory systems by addressing data integrity issues caused by hardware faults, environmental interference, and transient errors. Memory errors can be categorized into hard and soft errors. Hard errors are permanent memory faults caused by physical damage, manufacturing defects, or excessive heat, making affected memory cells unrecoverable and requiring hardware replacement or remapping techniques to maintain system reliability. Soft errors are temporary faults caused by radiation, voltage fluctuations, or charge leakage, leading to bit flips without permanent hardware damage [37]. Single-bit errors occur when only one bit in a memory word is flipped (e.g., $0 \rightarrow 1$ or $1 \rightarrow 0$) due to transient faults. These errors can be efficiently corrected using Hamming codes or Single Error Correction (SEC) codes. Multi-bit errors involve two or more bits flipping within the same memory word. Multi-bit errors require more advanced error correction codes (ECC), such as Double Error Correction (DEC) codes, Low-Density Parity-Check (LDPC) codes, or Reed-Solomon codes, to detect and correct faults effectively. Classical systems use well-established

techniques to detect and correct errors in memory hierarchies, cache subsystems, and data transmission.

1) TRADITIONAL ERROR DETECTION AND CORRECTION METHODS

Early methods such as parity checks and Hamming codes form the foundation of error detection and correction in classical computing. Hamming codes enable single-bit error correction and double-bit error detection, offering a low-complexity solution for systems like data storage and communication networks [38], [39]. Parity checks are often used to detect errors in cache hierarchies and data transmission.

2) ADVANCED ERROR CORRECTION TECHNIQUES

To improve error resilience in complex systems, advanced error detection and correction methods have been developed. In [40], authors focused on improving error detection and correction in fault-tolerant logic circuit design. They introduced a novel cross-code method that handled multiple errors with minimal area overhead, compared to traditional methods such as triple modular redundancy (TMR), LDPC, and Hamming codes. The method prioritized reducing the probability of uncorrected errors instead of correcting all errors, making it more efficient for high-density integrated circuits vulnerable to faults. Parity++ was presented in [41], which is an efficient error correction scheme for last-level caches. It provided Single Error Detection (SED) for all data and Single Error Correction (SEC) for key data, with 4x lower storage overhead and reduced energy compared to traditional ECCs. Two new error detection schemes for Multilevel Cell (MLC) memories were proposed in [42], which stored multiple bits per cell and were susceptible to limited magnitude errors. The One-Bit Parity (OBP) scheme detected magnitude-1 errors using a single parity bit, while the Two-Bit Parity (TBP) scheme detected both magnitude-1 and -2 errors with two parity bits. Both schemes simplified circuitry and improved efficiency compared to existing methods like Gray coding and Interleaved Parity.

Although classical error detection and correction techniques are highly effective for traditional memory systems, they face scalability challenges in high-performance computing and data-intensive applications. As memory architectures grow in complexity, achieving a balance between error resilience and computational efficiency becomes increasingly difficult. Advanced techniques, such as LDPC codes and hardware-accelerated ECC, are often used to enhance fault tolerance while minimizing performance overhead. However, as data volumes and processing demands continue to increase, further advances in error correction algorithms and hardware optimization are necessary to ensure reliable and efficient memory management in next-generation computing systems.

Recent advancements integrate machine learning for system-level anomaly detection in various domains such as signal integrity applications [43], aircraft sensor error detection [44], and EHR diagnostic error triggering [45] but its direct use in classical memory error detection remains limited. Future research could explore integrating ML methods with traditional error correction techniques to further enhance memory reliability.

D. PERFORMANCE METRICS

Performance metrics are essential to evaluate the efficiency and reliability of classical memory and storage systems. In memory systems, key metrics include latency, throughput, energy efficiency, and scalability, which collectively assess the effectiveness of memory hierarchies and management techniques.

Dynamic Random Access Memory is recognized for its low switching energy, which enables high throughput and bandwidth. However, its reliance on frequent refresh operations results in significant energy overhead. On the other hand, NAND achieves high storage density through multilevel cells but faces challenges, including higher switching energy and limitations in planar scaling. Retention, a key differentiator between volatile (DRAM) and non-volatile (NAND) memories, measures a memory cell's ability to maintain data integrity over time. While DRAM requires constant refreshing to retain data, NAND provides longer retention periods, making it more suitable for certain applications. Scalability remains a critical metric as both DRAM and NAND face structural constraints when scaling to higher-density applications [46].

For storage systems, performance is evaluated using metrics such as throughput, response time, capacity, reliability, and cost. Throughput measures the speed at which data is delivered, while response time evaluates the duration required to access data. Capacity assesses the storage system's limits, and reliability ensures the integrity of stored data. Cost is analyzed using composite metrics like capacity-per-dollar or throughput-per-dollar. Additionally, trade-offs between throughput and response time are often highlighted in composite metrics, showing how increased system utilization can improve throughput but degrade response time. Together, these metrics provide a comprehensive framework for optimizing the performance of memory and storage systems in modern computing environments [47].

IV. QUANTUM MEMORY MANAGEMENT

Memory management in quantum computing is fundamentally different from classical systems due to the unique characteristics of quantum information. Efficient memory management in quantum systems is necessary for preserving qubit states, ensuring stability and reliable quantum computations.

A. MEMORY ARCHITECTURE

Quantum memory is based on qubits, the fundamental units of quantum information. Qubits are typically organized into quantum registers, which act as temporary storage

locations during quantum computations. These registers allow quantum algorithms to manipulate multiple qubits efficiently by storing quantum information in a structured way. As described in [48], quantum register consisting of two qubits, $|q_1\rangle$ and $|q_0\rangle$, has a state defined as:

$$|q_R\rangle = |q_1\rangle \otimes |q_0\rangle = |q_1 q_0\rangle \tag{4}$$

where $\otimes$ denotes the tensor product, representing the combined quantum state.

Quantum registers play a crucial role in quantum processors, serving as an intermediate layer between quantum operations and memory. They enable essential quantum operations, such as superposition and entanglement, allowing computations to be executed on multiple quantum states simultaneously.

Figure 2 presents a high-level architectural overview of a Quantum Processing Unit (QPU), emphasizing the memory-related subsystems that interact with quantum registers and computation units. Similar to classical hierarchies where cache, RAM, and I/O interact with the CPU via a system bus, the QPU integrates specialized components such as quantum cache (Q-Cache), main quantum memory (QM), and QRAM. The Quantum Input/Output (QIO) and Quantum Network Interconnect (QNIC) interfaces enable data exchange with classical and quantum systems, respectively, while the Quantum Bus (Q-Bus) handles quantum communication within the QPU. Although our focus is not on QPU hardware design, this architectural model provides an intuitive parallel to classical memory hierarchies, helping to conceptualize how quantum memory is managed and structured.

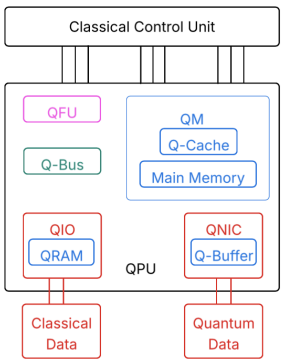

FIGURE 2. High-level functional architecture of a quantum processing unit [18], illustrating key components of quantum memory management such as the Quantum Functional Unit (QFU), Quantum Cache, main Quantum Memory, Quantum Bus, Quantum Input/Output interface, and Quantum Network Interconnect. Blue modules highlight quantum memory components, acting as the analog to hierarchical memory structure in classical architecture.

Quantum memory is designed using different types of qubits, including superconducting, neutral-atom, trapped-ion, photonic and spin qubits. Each type has unique characteristics that influence how quantum information is stored and processed.

1) SUPERCONDUCTING QUBITS

Superconducting qubits are built using LC oscillator circuits operated at cryogenic temperatures to minimize energy loss and maintain stability. The inductor is implemented as a Josephson Junction, which introduces non-linearity to the circuit and is typically made from superconducting materials like niobium or aluminum. The capacitor can be either an inter-digitated or parallel plate capacitor, and the Josephson Junction itself also contributes an intrinsic capacitance. Together, these components form a harmonic oscillator, where the lowest two energy levels are selected as the basic states of the qubit [49], [50].

2) NEUTRAL-ATOM QUBITS

Neutral-atom qubits store information in the hyperfine states of atoms confined in optical dipole traps. Individual atoms can be initialized, addressed, and manipulated using laser and microwave fields in the presence of magnetic field gradients. Their long coherence times, scalability, and compatibility with optical techniques make them a promising platform for quantum memory and logic operations [51].

3) PHOTONIC QUBITS

Photonic qubits leverage the quantum states of individual photons, offering a naturally mobile and low-noise system for quantum computing. Encoding is achieved through various degrees of freedom, including polarization, spatial modes, and time-bin encoding. Single-qubit operations are performed using interferometry with exceptionally high precision, while two-qubit gates are typically realized through linear optics combined with measurement-based techniques. Advances in integrated photonics, high-efficiency detectors, and scalable photon sources address key challenges such as photon loss and deterministic gate implementation [52].

4) TRAPPED-ION QUBITS

Trapped-ion qubits are implemented using atomic ions confined in electromagnetic traps, typically RF Paul traps, which enable long coherence times and precise control. The states of the qubit are defined by the internal electronic states of the ion, with transitions controlled via laser or microwave fields. Ions are confined in vacuum environments, minimizing decoherence and enabling stable quantum operations. The shared motional states of the ions allow for entanglement and controlled interactions, making them a promising platform for quantum computing [53].

5) SPIN QUBITS

Spin qubits use the spin states of electrons or nuclei to encode quantum information, typically implemented in semiconductor quantum dots (QDs) or donor atoms. They are classified based on spin encoding, including single-spin qubits (Loss-DiVincenzo), donor spin qubits (Kane's model), singlet-triplet qubits, and exchange-only qubits. The key control mechanism is the exchange interaction, which

enables quantum gate operations, while Zeeman splitting and spin-charge hybridization enhance manipulation and readout. Spin qubits benefit from long coherence times and compatibility with semiconductor fabrication, making them promising candidates for scalable quantum computing [54].

Beyond superconducting, neutral-atom, trapped-ion, photonic and spin qubits, several other qubit technologies are being explored for quantum memory and computation. Topological qubits utilize anyons in topologically ordered system. Topological quantum computing is based on properties of topological phases of matter and is promising approach for fault-tolerant quantum computing [55].

Qubit decoherence is a primary challenge in quantum memory management. Decoherence occurs when qubits interact with their environment, leading to a loss of quantum information [56]. Decoherence can be influenced by noise and external interference. Decoherence times measure the stability of qubits and directly impact the reliability of quantum memory. We will discuss more about quantum noise and decoherence in section VI which are major challenges we need to overcome to achieve fault-tolerant quantum computing.

B. MEMORY ACCESS AND ALLOCATION

Quantum memory access differs significantly from classical systems, as it depends on quantum gates and circuit-based operations rather than address-based access mechanisms in classical memory. Unlike classical systems where memory is statically or dynamically allocated based on addresses, quantum systems allocate qubits dynamically according to the requirements of the quantum circuit design. Efficient allocation strategies are essential to minimize the impact of decoherence and error propagation during computation. Techniques such as error correction codes, noise mitigation, and structured architectures, such as Quantum Random Access Memory, are crucial in optimizing qubit allocation and utilization.

1) QUANTUM GATES

Quantum gates perform operations on qubits, similar to classical logic gates (e.g., AND, OR, NOT), but with the unique capability to handle quantum properties such as superposition and entanglement. Common quantum gates include Pauli gates, Hadamard gates, Controlled-NOT (CNOT) gates, and Phase Shift gates [57]. These gates manipulate qubits through unitary transformations, enabling complex computations and the execution of quantum algorithms.

The Pauli gates are fundamental quantum gates that manipulate qubits. The Hadamard gate creates superposition, transforming a qubit $|0\rangle$ into $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|1\rangle$ into $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$. The CNOT gate entangles two qubits by flipping the target qubit when the control qubit is $|1\rangle$, a key operation for creating entangled states. The Phase Shift gate introduces a phase ϕ to the $|1\rangle$ state while leaving $|0\rangle$ unchanged. Toffoli gate is also known as the controlled-controlled-NOT (CCNOT) gate. It consists of two control qubits and one target qubit.

These gates enable quantum parallelism, entanglement generation, and phase manipulation, providing the foundation for implementing complex quantum circuits and algorithms [58]. They not only perform computations but also have a critical role in memory management, as each gate operation affects qubit states stored in quantum registers. Unoptimized gate usage can accelerate decoherence, which leads to memory loss in quantum architectures.

2) QUANTUM RANDOM ACCESS MEMORY

Quantum Random Access Memory enables superposition-based data access, allowing quantum algorithms like Grover's search and quantum machine learning algorithms to efficiently process large datasets by leveraging quantum parallelism.

The bucket-brigade QRAM model provides favorable error scaling $O(plog^2N)$ but faces scalability, precision, and error propagation challenges [59], [60]. QRAMpoly uses a polynomial encoding of binary strings to achieve an exponential improvement in T-depth- reducing it from $O(logN)$ (as seen in bucket-brigade model) to $O(loglogN)$. This also lowered T-count to $O(N-logN)$ ($O(N)$ in bucket brigade) while maintaining an $O(N)$ qubit count [61]. Fan-Out QRAM uses qubits where each kth address qubit controls 2^k quantum switches. All the quantum switches are initialized to $|0\rangle$, activating paths based on input addresses. While enabling simultaneous access to multiple memory locations, its exponential switch requirement $O(2^n)$ makes large-scale implementation impractical [62], [63].

Flip-Flop QRAM (FF-QRAM) is a quantum circuit-based QRAM that stores binary data in superposition through a three-stage process: flip stage, register stage, and flop stage. While it reduces hardware overhead, it requires an exponential circuit depth $O(2^n)$ as well, which limits its feasibility for deep quantum circuits [65]. Qudit-based QRAM utilizes higher-dimensional quantum states (qudits) to compress memory storage, reducing ancilla requirements [66]. However, unstable higher states result in higher error rates, making their implementation challenging. EQGAN QRAM leverages entanglement-based quantum GANs for efficient data encoding with constant-depth circuits [67], while Approximate PQC-Based QRAM sequentially stores classical data using trainable parameterized quantum circuits, improving structured dataset storage but lacking parallel access benefits [68]. Fat-Tree QRAM is a scalable, pipelined architecture that enables parallel quantum queries with enhanced bandwidth and reduced latency compared to traditional QRAM designs [69]. It pipelines $O(log(N))$ independent queries in $O(log(N))$ time using $O(N)$ qubits, enabling high-bandwidth parallel access to data. As illustrated in Figure 3, the evolution of QRAM architectures has progressed from early designs, like Fan-Out QRAM and

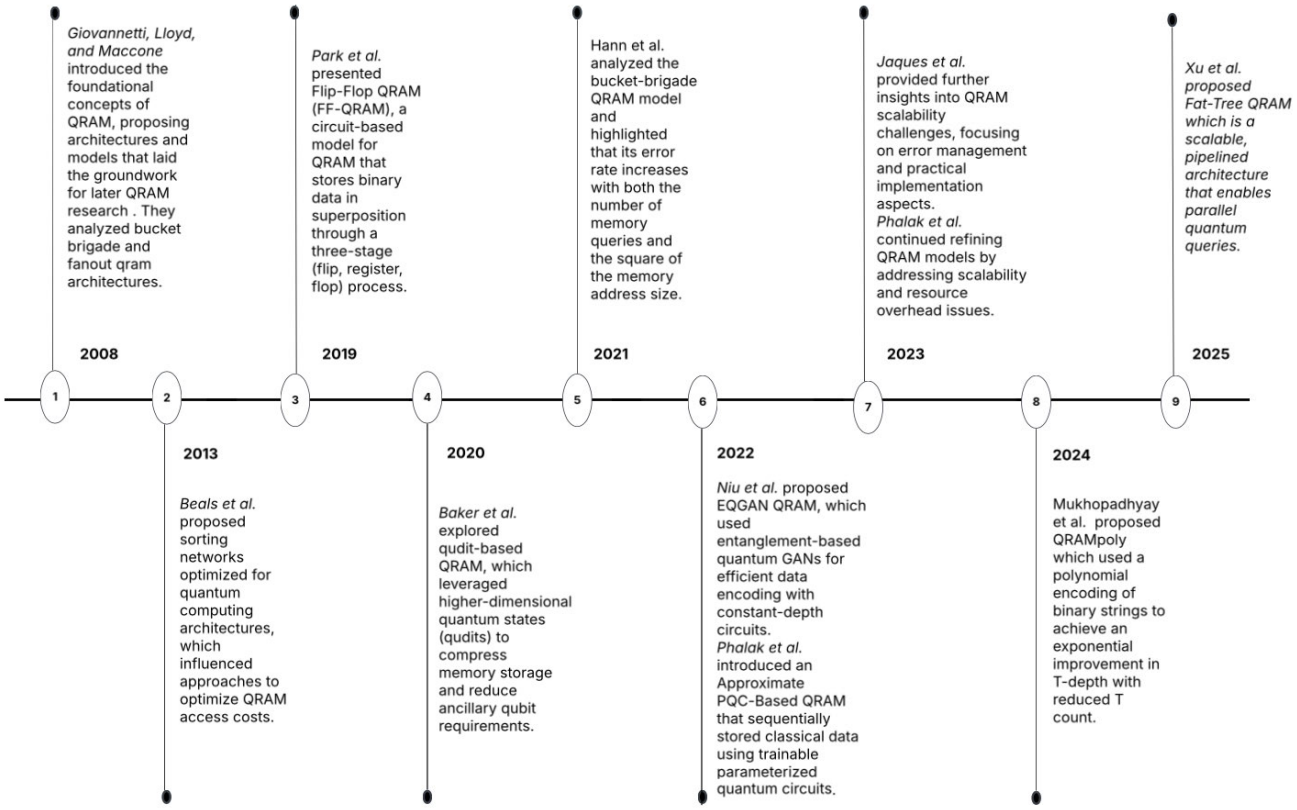

FIGURE 3. Advancements and evolution of QRAM.①: [63], ②: [64],③: [65],④: [66], ⑤: [59],⑥: [67], [68],⑦: [60], [62],⑧: [61],⑨: [69].

bucket-brigade QRAM, to more advanced implementations such as Fat-Tree QRAM.

Sorting networks optimize access cost to $O((N+k)log(N+k))$ under a hypercube architecture, but their feasibility is uncertain [64]. Active QRAM requires costly external interventions (e.g., lasers in trapped-ion systems), while passive QRAM faces difficulties in building scalable Hamiltonians [63].

Different technological approaches face unique challenges:

- Trapped Ions QRAM: Limited by photon-ion interaction fidelity [63].
- Transmon-Based QRAM: Scaling beyond one layer is constrained by relativistic causality bounds [70].
- Photonic Heralded QRAM: Requires complex collective measurements [71].

When compared to classical RAM with $O(logN)$ active components, QRAM requires $\Omega(N)$ interventions per access, making it resource-intensive [72]. While QRAM remains very important for quantum computing, practical scalability demands advancements in hardware connectivity, error correction, and passive memory design [60].

QRAM has a foundational role in quantum algorithms that achieve acceleration through efficient data access. Applications such as Grover's search, quantum nearest neighbor classification, quantum support vector machines, and quantum principal component analysis rely on QRAM to load and query large classical datasets in superposition, enabling speedups unattainable with classical memory. These use cases highlight QRAM's importance in advancing scalable quantum machine learning and search-based computations.

C. ERROR DETECTION AND CORRECTION

Quantum memory is susceptible to multiple error sources. Bit-flip errors change the state of a qubit (e.g., $|0\rangle$ to $|1\rangle$ or vice versa). Phase-flip errors change the relative phase of a qubit in superposition. Measurement errors include inaccuracies during state readout [73]. These errors degrade the fidelity of quantum memory, making robust error correction critical for stable operations. Quantum Error Correction codes are essential to solve challenges with errors in quantum systems. Unlike classical error correction, QEC protects fragile quantum states without directly measuring them which would collapse the quantum state.

1) SHOR'S CODE

Shor's code is a Quantum Error Correction method that encodes one logical qubit into nine physical qubits to correct both bit-flip and phase-flip errors [74], [75].

The target qubit is the main qubit, while the eight supporting qubits are designated as ancillary qubits (1 to 8). The encoding process begins by replicating the main qubit's state into two additional qubits (3rd and 6th qubits) using

CNOT gates, enabling phase error correction. These qubits are then put into superposition using Hadamard gates to prepare them for subsequent error detection.

To protect against bit-flip errors, each of these three qubits further transfers its state to two ancillary qubits, ensuring redundancy:

- Main qubit transfers its state to the 1st and 2nd qubits.
- 3rd qubit transfers its state to the 4th and 5th qubits.
- 6th qubit transfers its state to the 7th and 8th qubits.

After the encoding phase, potential errors (bit-flip or phase-flip) may occur denoted as E. The correction process involves applying CNOT gates and Hadamard gates in reverse order to detect the error. Toffoli gates, controlled by the ancillary qubits, are applied to the main, 3rd, and 6th qubits to correct phase errors. Hadamard gates are then used to bring the qubits out of superposition. Finally, a CNOT gate and a Toffoli gate, controlled by the 3rd and 6th qubits, ensure the final error correction.

2) STEANE'S CODE

Steane's code is a Quantum Error Correction scheme that extends classical linear error-correcting codes into the quantum domain. Based on the [7,4,3] Hamming code, it encodes one logical qubit into seven physical qubits, providing fault tolerance by detecting and correcting both bit-flip (X) and phase-flip (Z) errors.

This code utilizes stabilizer measurements to identify errors without directly disturbing the encoded quantum information. By applying quantum parity checks on both the computational and Hadamard-transformed bases, Steane's code ensures simultaneous protection against X and Z errors. This structure allows quantum superpositions to remain intact, a crucial requirement for maintaining coherence in quantum computation.

As one of the earliest examples of stabilizer codes, Steane's code forms the foundation for modern Quantum Error Correction, connecting block codes like Shor's code and large-scale stabilizer codes like the Surface code. The ability to redundantly encode quantum states while preserving coherence established a critical step toward practical fault-tolerant quantum computation [76].

3) TOPOLOGICAL CODES

Topological codes are a class of quantum error-correcting codes that use topological properties to encode and protect quantum information from errors. The toric code is a topological quantum error-correcting code defined on a 2D lattice with periodic boundary conditions [77]. It has high error threshold and is robust against local errors but requires complex decoding algorithms.

Surface codes are also defined in a two-dimensional lattice, but unlike the toric codes, they don't have periodic boundary conditions. Qubits are classified into data qubits and measurement qubits. Data qubits store computational states while measurement qubits detect and correct errors. Measurement qubits are further divided into measure-Z and measure-X qubits, responsible for enforcing stabilizer constraints by projecting adjacent data qubits into eigenstates of multi-qubit Pauli-Z and Pauli-X operators, respectively. This stabilizer-based approach enables error detection and correction without directly measuring the data qubits, preserving quantum coherence. Surface codes require fundamental quantum operations, including initialization, single-qubit rotations, nearest-neighbor CNOT gates, and SWAP operations to exchange quantum states between data and measurement qubits. By continuously performing stabilizer measurements, the surface code ensures fault-tolerant quantum computation with scalable error correction [15]. Surface codes correct both incoherent and coherent noise. At large scales, coherent physical noise is effectively transformed into random Pauli errors at the logical level, simplifying error correction [78]. Moreover, they enable exponential suppression of logical errors with increasing code distance and effectively mitigate leakage errors using auxiliary qubit resets, reinforcing their viability for large-scale fault-tolerant quantum computation [79]. In [80], authors demonstrated Quantum Error Correction operating below the surface code threshold using superconducting qubits. The authors implemented distance-5 and distance-7 surface codes on new-generation processors – Willow, achieving logical error suppression with scaling factor $\Lambda \approx 2.14$, which surpassed the performance of individual physical qubits. A real-time decoder was integrated, maintaining below-threshold performance across up to one million cycles with a fast 1.1 μs cycle time and 63 μs decoding latency. They also run distance-29 repetition codes and observe ultra-low logical error rates down to 10^{-10}, revealing rare correlated error events as a limiting factor. These results mark a significant milestone toward scalable, fault-tolerant quantum computation.

3D color codes are a class of topological quantum error-correcting codes that extend 2D color codes to three-dimensional lattices, enabling transversal implementation of the full Clifford group and offering improved fault tolerance for scalable quantum computation [81].

4) ADVANCED QUANTUM ERROR CORRECTION

Beyond fundamental Quantum Error Correction techniques like Shor's code, Steane's code, and Topological codes, recent advancements have introduced more modular and resource-efficient approaches for quantum fault tolerance.

The Bacon-Shor code is a subsystem quantum error-correcting code that encodes logical qubits into a rectangular lattice of physical qubits. It leverages gauge degrees of freedom to simplify error detection and correction by using local row and column parity checks. This structure reduces the complexity of syndrome extraction, making it efficient for fault-tolerant quantum computation [82].

Another notable approach is Haah's Cubic code, a three-dimensional local stabilizer code that does not have

string-like logical operators, making it a promising candidate for self-correcting quantum memory [83]. Unlike conventional topological codes, Haah's code uses fractal-like stabilizers, which prevent errors from propagating easily, thereby increasing the energy barrier for logical errors. This property makes it resistant to thermal noise. Haah's code is effective for long-term quantum information storage.

The Heavy-Hexagon and Heavy-Square codes optimize Quantum Error Correction for superconducting qubits by reducing frequency collisions while maintaining fault tolerance. They use flag qubits to detect and correct errors efficiently. A new decoding algorithm ensures full-distance error correction. The heavy-hexagon code combines elements of surface and Bacon-Shor codes, while the heavy-square code modifies the surface code. Both achieve high error thresholds, making them practical for scalable quantum computing [84].

Quantum Low-Density Parity-Check (QLDPC) codes provide efficient error correction by distributing stabilizers sparsely across qubits, enabling fault tolerance with fewer physical resources. These codes offer scalable error correction while minimizing the overhead required for logical qubit encoding [85]. Surface codes are often considered a special subclass of QLDPC codes due to their sparse, local stabilizer structure, even though they are also categorized as topological codes. In [86], authors demonstrated a constant-overhead fault-tolerant quantum computing architecture by leveraging quantum low-density parity-check codes implemented on reconfigurable neutral atom arrays. The authors proposed a modular, scalable architecture in which logical qubits are encoded using QLDPC codes with constant rate and linear distance scaling. They show how long-range connectivity in neutral atom systems enables constant-time syndrome extraction through parallel operations, and introduce QLDPC lattice surgery for logical gate operations. Their approach achieves both constant space-time overhead and physical feasibility, marking a significant step toward scalable quantum computation. In [87], authors presented a practical and high-performance fault-tolerant quantum memory architecture based on a new family of low-density parity-check codes called Bivariate Bicycle (BB) codes. Unlike traditional surface codes, BB codes offer high encoding rates, linear distance scaling, and $10\times$ lower qubit overhead. Specifically, the authors demonstrated that 12 logical qubits can be preserved for nearly 1 million syndrome cycles using just 288 physical qubits, compared to approximately 3,000 qubits required by surface codes. They introduced a depth-7 syndrome measurement circuit, adapted a belief-propagation decoder for realistic circuit-level noise, and showed that their codes achieve an error threshold of approximately 0.7%, matching that of the surface code. The Tanner graphs of BB codes are decomposed into two planar layers (thickness-2), making them feasible for implementation on superconducting hardware. This work provides the way for near-term realization of scalable and efficient quantum memories. In [88], authors proposed a Belief Propagation with Ordered Statistics Decoding (BP-OSD) method to enhance the decoding performance of degenerate QLDPC codes under depolarizing noise. The approach combined soft-decision decoding (belief propagation) with a post-processing step based on ordered statistics decoding, enabling substantial error correction improvements. The authors demonstrated that BP-OSD outperformed traditional decoders across several known and newly constructed QLDPC codes, including generalized bicycle and generalized hypergraph product codes – some of which even surpassed large surface codes decoded with near-optimal algorithms.

Recent advancements in Quantum Tanner codes enhance Quantum Error Correction by achieving constant rate and linearly growing minimum distance. Derived from classical Tanner codes and expander graphs, they construct quantum codes with superior minimum distance estimates and efficient error detection. Unlike traditional QLDPC codes, Quantum Tanner codes leverage locally testable codes (LTCs) for minimal qubit overhead [89].

In [90], authors introduced a Reinforcement Learning (RL)-based approach for optimal Quantum Error Correction codes using the Quantum Lego (QL) framework. Many-hypercube codes was introduced in [91], a new family of quantum error-correcting codes that achieved higher encoding rates compared to surface codes and QLDPC (hypergraph product) code. These codes enable efficient parallel logical gate operations while maintaining fault tolerance. In this study, authors also introduced level-by-level minimum distance decoding and fault-tolerant encoders to improve efficiency. Many-hypercube codes are promising alternative for scalable, high-performance quantum computing.

Error detection and correction have a vital role in maintaining quantum memory. It enhances reliability by correcting errors before they propagate which increases coherence time and provides high scalability for quantum systems. Error correction is essential for building large-scale quantum systems [56]. Without effective error correction, quantum memory would be unstable for practical applications. A detailed overview of the various Quantum Error Correction codes available in the existing literature is provided in table 1.

D. QUANTUM READ/WRITE OPERATIONS

Quantum read/write operations involve encoding, manipulating, and retrieving quantum information from qubits. These operations are fundamental to quantum memory stability and error correction, directly impacting the efficiency of quantum computation. Advancements in quantum state measurement and control have significantly improved readout fidelity, initialization speed, and coherence preservation. Research efforts have focused on various quantum platforms, including superconducting, trapped ions, photonic, neutral atoms, and silicon-based spin qubits. We explored the readout and initialization techniques for these different qubit platforms.

TABLE 1. Quantum error correction techniques.

Error Correction Technique	Code Family	Encoding Size	Key Features	Scalability & Feasibility
Shor's code [74]	Concatenated	9 qubits	Encodes one logical qubit into nine physical qubits using a concatenated 3-qubit repetition scheme for phase-flip and bit-flip error correction; early example of Quantum Error Correction.	High qubit overhead; not very resource-efficient.
Steane's code [76]	Calderbank-Shor-Steane (CSS) code	7 qubits	Based on classical Hamming [7,4,3] code; stabilizer-based detection and correction; fault-tolerant gates.	More resource-efficient than Shor's Code.
Toric code [77]	Topological	Varies(2D Lattice-based)	A topological code that encodes logical qubits in global properties of a 2D lattice with periodic boundary conditions.	Robust against local errors; requires complex decoding algorithms.
Surface code [15]	QLDPC (Topological)	Varies(2D Lattice-based)	Uses stabilizers for error detection and correction; high fault tolerance; nearest-neighbor interactions.	Highly scalable and high error threshold, well-studied, many efficient decoders.
3D color codes [81]	Topological	Varies(3D Lattice-based)	Supports transversal Clifford operations, improving fault tolerance; operates on a three-dimensional lattice.	Suitable for large-scale quantum computing but requires more physical qubits.
Bacon-Shor code [82]	Subsystem code	Varies(Rectangular lattice)	Uses gauge operators to simplify error detection; reduces complexity of syndrome extraction.	Efficient fault tolerance for superconducting and trapped-ion qubits.
Haah's Cubic code [83]	Topological (Fractal)	Varies(3D lattice)	Fractal-like stabilizers prevent long-range error propagation; potential for self-correcting quantum memory.	Promising for long-term quantum information storage; implementation is still theoretical.
Heavy-Hexagon code [84]	Hybrid (Topological + Subsystem)	Varies(Heavy-hexagonal lattice)	A hybrid of Surface and Bacon-Shor codes, designed for superconducting qubits; reduces frequency collisions.	Practical for superconducting qubits; supports high-fidelity quantum operations.
Bivariate Bicycle (BB) codes [87]	QLDPC	Varies (LDPC-based planar layout)	Family of QLDPC codes with high encoding rates, linear distance scaling, and low qubit overhead; uses belief propagation decoder on planar Tanner graphs.	Preserves 12 logical qubits over 1 million cycles using only 288 qubits; 10 times more efficient than surface codes; feasible on superconducting platforms.
QLDPC codes on Atom Arrays [86]	QLDPC	Varies (Neutral atom lattice)	Constant-rate, linearly-scaling QLDPC codes using modular reconfigurable neutral atom arrays; supports lattice surgery.	Enables constant-time syndrome extraction and fault tolerance with constant overhead.
BP-OSD Enhanced QLDPC codes [88]	QLDPC	Varies (Generalized Bicycle/Hypergraph codes)	Combines belief propagation with ordered statistics decoding to improve performance under depolarizing noise.	Outperforms traditional decoding and some large surface codes.
Quantum Tanner codes [89]	QLDPC	Varies (Expander Graph-based)	Uses locally testable codes (LTCs) to achieve high fault tolerance with efficient stabilizer checks.	Offers high fault tolerance with minimal qubit overhead.
Many-Hypercube codes [91]	QLDPC	Varies(Hypercube-based structure)	Supports parallel logical gate operations while maintaining fault tolerance; improves encoding rates.	Higher encoding rates than Surface and QLDPC (hypergraph product) code; suitable for parallel processing but requires efficient decoding algorithms.

1) SUPERCONDUCTING QUBITS

Superconducting qubits are one of the leading platform for quantum computation, requiring precise readout techniques. Several studies have explored different methods to improve readout fidelity and measurement speed. In [92], a novel readout scheme for superconducting qubits was proposed which combined a shelving technique to minimize decay error during readout and two-tone excitation of the readout resonator to distinguish higher-energy qubit states. Machine learning algorithms further enhanced qubit-state assignment fidelity, improving single-shot frequency-multiplexed qubit readout. The study achieved 99.5% assignment fidelity for two-state readout and 96.9% for three-state readout, without using a quantum-limited amplifier. With a 140 ns readout time, this approach significantly improved quantum state measurement accuracy. In [93], authors demonstrated optical readout of a superconducting transmon qubit with a piezo-optomechanical transducer, converting microwave signals into optical signals for readout through optical fiber. This approach reduced cryogenic constraints, improving scalability in quantum processors. The method achieved 81% single-shot readout fidelity, highlighting the potential of microwave-to-optics conversion for efficient quantum computing. An efficient qubit initialization protocol using a tunable dissipative environment was proposed in [94] to achieve fast and accurate qubit state preparation. The protocol used a superconducting qubit in interaction with a thermal bath with a tunable harmonic oscillator, dynamically adjusting the environment to induce rapid relaxation to the ground state. Using a Markovian master equation, authors

optimized initialization parameters and demonstrated that qubit initialization speed and fidelity can be significantly improved. Numerical results confirmed that the protocol achieved a high-fidelity ground state while minimizing excess dissipation, making it suitable for large-scale quantum computing. A quantum optimal control approach for single-shot multi-qubit gates was explored in [95], achieving fidelities above 99.99%. Using a feasibility-based formulation and global optimization techniques, the method improved the speed and accuracy of three- and four-qubit gates without decomposition into smaller gates, which reduced decoherence effects. The approach was tested on superconducting transmons using avoided-level-crossing-based control, demonstrating faster and more efficient gate operations. This study provided a scalable solution for high-fidelity quantum computing, making advanced quantum control techniques more accessible.

2) NEUTRAL-ATOM QUBITS

Neutral-atom qubits provide a scalable approach to fault-tolerant quantum computing due to their high coherence times and the ability to perform non-destructive readout. In [96], authors demonstrated a universal neutral-atom quantum computer with individual optical addressing and non-destructive readout. The system achieved 99.35(4)% CZ fidelity (where the number in parentheses indicates the uncertainty, i.e., 99.35 $\pm$ 0.04%) and 99.902(8)% local single-qubit RZ gate fidelity, supporting scalable fault-tolerant quantum computing. By using steerable laser beams instead of mid-circuit qubit shuttling, high gate rates limited only by optical switching times were achieved. Additionally, the non-destructive readout method achieved a state-averaged atom loss rate of 0.9(3)%, allowing qubit reuse and enhancing computational efficiency. In [97], authors demonstrated high-fidelity parallel entangling gates with 99.5% fidelity across up to 60 qubits, advancing neutral-atom quantum computing. They used Rydberg interactions to implement two-qubit controlled phase (CZ) gates and extended the technique to three-qubit gates. By optimizing laser pulse control, suppressing scattering errors, and improving atom cooling, they achieved significant error reduction. Multiple benchmarking methods confirmed the fidelity, and scalability was demonstrated with consistent performance across a large qubit array.

3) PHOTONIC QUBITS

Photonic qubits offer unique advantages in quantum information processing due to their inherent low decoherence, high-speed transmission, and compatibility with optical fiber networks. In [98], authors presented a programmable photonic quantum memory capable of storing 72 optical qubits using 144 atomic ensembles, supporting up to 1,000 consecutive read/write operations. The system demonstrated quantum queue, stack, and buffer functionalities, along with the storage and reshuffling of entangled photon pairs, enabling efficient quantum networking and repeater protocols. Utilizing 87Rb atomic ensembles and Electromagnetically Induced Transparency (EIT), the memory achieved 95% fidelity, a 500 μs coherence time, and a 1 μs access speed.

However, photon transmission over long distances presents fundamental challenges due to exponential photon loss in optical fibers, which limits direct quantum communication to approximately 500 km [99]. To overcome this, quantum repeaters leverage entanglement and quantum teleportation to extend communication distances, requiring high-fidelity quantum memories as demonstrated in [98]. Additionally, heralded qubit amplifiers, which probabilistically amplify the probability amplitude of photonic qubits, offer a promising alternative for Device-Independent Quantum Information Processing (DIQIP) over tens of kilometers [99]. These advances contribute to the ongoing development of scalable and robust quantum communication networks.

4) TRAPPED-ION QUBITS

Trapped-ion quantum computing benefits from long coherence times and precision control, making them a potential candidate for scalable quantum systems. High-fidelity qubit operations in a trapped-ion quantum computing system using $^{43}Ca^{+}$ hyperfine atomic clock states was demonstrated in [100]. In this study, authors achieved 99.93% state preparation and readout fidelity, a coherence time of 50 seconds, and an average single-qubit gate fidelity of 99.9999%. The ion qubit was trapped in a room-temperature microfabricated surface trap without the need for magnetic shielding or dynamic decoupling. The system utilized near-field microwave control, eliminating the need for lasers for logic gates, making it scalable for large arrays of multiplexed ion traps. In [101], authors used a two-layer μ-metal shield and ring-shaped Sm_2Co_{17} magnets (0.37 mT) to stabilize a single $^{40}Ca^{+}$ Zeeman qubit. They achieved a Ramsey coherence of $T_2^* \approx 300$ ms and a spin-echo coherence of $T_2 \approx 2.1$ s. Authors in [102] stored a qubit in the metastable $5D_{5/2}$ manifold of a single $^{137}Ba^{+}$ ion. With sympathetic cooling and erasure error detection, they achieved a CPMG spin-echo coherence of $T_2 = 136(42)$ s whereas the same qubit without erasure detection dephased in 22(2) s.

5) SILICON-BASED SPIN QUBITS

Silicon-based spin qubits have emerged as a promising solution for scalable quantum computing due to their long coherence times and compatibility with existing semiconductor technology. Recent advancements in high-fidelity readout and initialization techniques have significantly improved their reliability, making them important candidates for fault-tolerant quantum systems. In [103], high-fidelity readout and control of a 31P nuclear spin qubit in silicon was demonstrated, which achieved higher than 99.8% readout fidelity and single-qubit gate fidelities above 98%. Researchers integrated single-shot electron spin readout with coherent NMR control, enabling quantum non-demolition

(QND) measurement. The nuclear spin exhibited coherence times of 60 ms, and an on-chip electrical detection method facilitated its manipulation. These results validated the feasibility of silicon-based nuclear spin qubits for scalable quantum computing. In [104], authors demonstrated high-fidelity readout and initialization of silicon spin qubits above 1K. They used radiofrequency readout for fast and accurate measurements, achieving a readout fidelity of 99.34%. To counter thermal effects at higher temperatures, they introduced an algorithmic initialization protocol, enabling deterministic state preparation despite thermal broadening. The results achieved were 99.85% single-qubit gate fidelity and 98.92% two-qubit gate fidelity, demonstrating reliable read and write operations essential for scalable quantum computing.

Quantum read/write operations also depend on pulse optimization techniques to minimize noise, enhance gate fidelities, and improve computation reliability and quantum memory efficiency. A pulse optimization framework to improve gate fidelities for semiconductor spin qubits was presented in [105]. Authors addressed challenges such as charge noise, bandwidth limitations, and crosstalk by adapting spectral concentration techniques for pulse shaping. They achieved fidelities above 99.5% for two-qubit gates, with numerical simulations predicting higher than 99.9% fidelity in short durations.

Quantum read/write operations are crucial for ensuring the accuracy and efficiency of quantum computations. Advances in superconducting, neutral-atom, trapped-ion, and silicon-based qubits have led to significant improvements in readout fidelity, state initialization, and control techniques. These advancements push the boundaries of scalability, allowing for more reliable implementations of fault-tolerant quantum computing. As quantum hardware continues to evolve, integrating machine learning and optimal control methods will further enhance the precision and robustness of quantum information processing.

A comparative summary of key performance metrics across leading quantum memory technologies is provided in Table 2, highlighting differences in coherence times, fidelity, scalability, and energy efficiency. Figure 4 provides the trade-off between coherence time and gate (or memory) fidelity for the major qubit technologies.

E. PERFORMANCE METRICS

Quantum memory management depends on several key performance metrics that determine computational reliability and efficiency:

- Coherence Time: Measures how long a qubit retains its quantum state before decoherence. Longer coherence times enable more reliable execution of complex quantum algorithms.
- Fidelity: Quantifies the accuracy of quantum gate operations by assessing how closely the actual output matches the intended state. Gate fidelities above 0.99 are crucial for ensuring dependable quantum circuits. Additionally, fidelity-based coherence measures provide analytical insights into how quantum states deviate from incoherent states [108], contributing to circuit stability and optimization.
- Scalability: Remains a major challenge. As qubit counts increase, longer circuits introduce additional crosstalk, gate errors, and readout noise, leading to error accumulation. Efficient error mitigation techniques are necessary to maintain computational accuracy.
- Decoherence Effects: Includes energy relaxation (T_1) and partial wavefunction collapse (T_1^*), both of which significantly impact algorithm success rates, particularly in larger circuits.

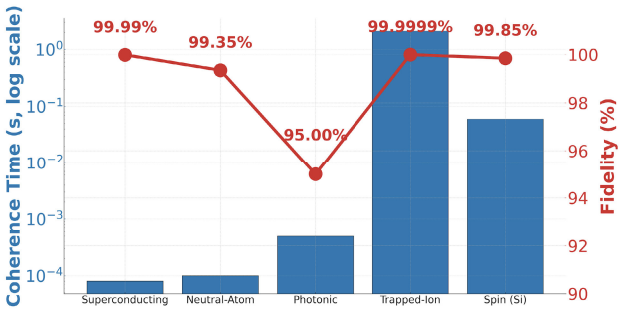

FIGURE 4. Coherence vs. fidelity for leading qubit platforms. Bars show the longest routinely attainable energy-relaxation time T_1, except for trapped ions where the published single-echo T_2 (2.1 s) is used. Red points mark the highest reported two-qubit gate fidelity; for photonic memories, the best read/write fidelity (95 %) is plotted. The logarithmic axis highlights the 10^{-4} to 10^{0} s span in coherence.

These performance metrics collectively guide the design and optimization of quantum systems and algorithms, helping address the challenges of noisy qubits and improving scalability in quantum computation [109].

V. COMPARATIVE ANALYSIS OF MEMORY MANAGEMENT

Memory management in classical and quantum computing differs significantly due to their distinct architectures and operational principles. Understanding these differences is crucial for optimizing performance in each paradigm. Classical memory systems use bits stored within structured levels of registers, cache, RAM, and secondary storage [28]. Data management strategies such as paging and segmentation enable efficient allocation and retrieval of resources [68]. On the other hand, quantum memory encodes information using qubits, which leverage superposition and entanglement for simultaneous representation of multiple states [24]. However, preserving these quantum states is a key challenge in quantum memory systems [110].

Classical registers store binary data and operate deterministically, supporting fast and reliable access by the CPU. On the other hand, quantum registers store quantum states using multiple qubits, enabling simultaneous manipulation

TABLE 2. Performance comparison of quantum memory technologies.

Qubit Type	Coherence Time	Read/Write Fidelity	Scalability	Energy Efficiency
Superconducting	80 µs (T_1) [106]	99.5% readout (2-state) [92]; 99.99% gate fidelity [95]	High gate speed; microwave-optical transducer improves scalability [93]	High cooling cost; active control needed [94]
Neutral-Atom	∼100 µs (Rydberg T_1) [107]	99.35% CZ, 99.902% RZ gate fidelity [96]	Scalable via optical addressing; supports up to 60 qubit parallelism [97]	Requires atom cooling; laser-based control
Photonic	∼500 μs (EIT-based) [98]	95% fidelity, 1000 read/write ops [98]	Excellent for quantum networks; limited by photon loss [99]	No active cooling; minimal idle energy consumption
Trapped-Ion	2.1 s (spin-echo T_2 [101]	99.9999% gate fidelity; 99.93% R/W [100]	High precision; supports scalable multiplexed trap arrays [100]	Moderate power; vacuum and laser systems required
Spin (Silicon)	∼60 ms [103]	99.34% readout; 99.85% gate fidelity [104]	High CMOS compatibility; operable above 1K [104]	Excellent; low energy operation possible [103]

of superposed values through entanglement. While classical registers are stable and straightforward to manage, quantum registers require precise control and error correction due to their susceptibility to decoherence and noise. These differences reflect the broader distinctions in memory management approaches between classical and quantum architectures.

A. ERROR CORRECTION TECHNIQUES

Classical memory systems depend on robust error detection and correction mechanisms to maintain data integrity and system reliability. Techniques such as Hamming codes, Low-Density Parity-Check codes, and Error Correction Codes address bit-flips and transient errors caused by hardware faults, electromagnetic interference, and memory wear-out [3], [68].

Quantum memory, on the other hand, introduces unique challenges due to quantum decoherence and noise, which are caused by environmental interactions and disrupt qubit states [13]. Unlike classical errors, quantum errors can involve both bit-flip and phase-flip errors, requiring advanced Quantum Error Correction techniques to preserve quantum information [14], [76]. A 2.5 year field study reported DRAM bit-flip rates of $\approx 2.5 \times 10^{-11}$ failures bit^{-1} h^{-1} (equivalently 25 000–70 000 FIT Mbit^{-1}), with 8% of DIMMs with correctable errors and 0.22% showing uncorrectable errors per year [111]. However, the state-of-the-art operation-time errors in quantum processors remain much higher. Authors in [112] used tunable-coupling gmon qubits to realize the full two-parameter fSim gate set and benchmarked 525 distinct gates, achieving an average two-qubit Pauli error of 3.8×10^{-3} across the entire (θ, ϕ) space. In [113], authors achieved microwave-driven single-qubit gates on a $^{43}Ca^{+}$ clock qubit with an error of 1.5×10^{-7} in a room-temperature surface trap via active amplitude/phase calibration and suppression of decoherence and leakage. These differences make it clear why classical ECC can correct with modest overhead, whereas quantum systems require heavy-weight codes to reach fault-tolerant thresholds.

QEC codes such as Shor's code, Steane's code, and the Surface code encode a single logical qubit into multiple physical qubits to detect and correct errors without collapsing the quantum state [15], [78]. However, these techniques introduce significant computational overhead, as QEC requires thousands of physical qubits for every logical qubit to ensure fault tolerance [22]. Authors in [114] implemented a distance-5 surface code on 49 physical qubits and achieved a 2.9 % logical error per cycle – which outperformed ensemble of distance-3 logical qubits on average. Authors in [80] ran a distance-7 code – 101 physical qubits delivered a 0.143 % logical error per cycle and a lifetime 2.4 times longer than its best physical qubit, marking the first below-threshold surface-code memory. These results highlight that pushing to a larger code distance initially increases the physical-qubit overhead (101 vs 49). But, by moving the processor firmly below threshold, this reduces the logical-error rate by nearly 20 times. This ensures that future hardware-fidelity gains will let the same target error be reached with smaller distances – so the apparent penalty in storage is the price of unlocking exponential suppression of logical errors (see Table 3).

Unlike classical error detection and correction, QEC cannot depend on direct duplication of data due to the no-cloning theorem, which prevents exact copying of quantum states [115]. Instead, QEC uses stabilizer measurements and syndrome extraction to detect and correct errors while preserving superposition and entanglement. Surface codes provide exponential suppression of logical errors with increasing code distance [79]. However, large-scale implementation of QEC remains an active research challenge due to hardware constraints, gate fidelity limitations, and qubit connectivity issues [116].

While both classical and quantum memory systems depend on error correction techniques to enhance reliability, their methodologies differ significantly. Advancing both paradigms require ongoing research to optimize error correction efficiency, reduce computational overhead, and improve scalability for the next generation of high-performance and quantum computing architectures [13], [116].

B. LATENCY, ENERGY AND SCALABILITY

Classical and quantum memory systems face different challenges at the hardware and architectural levels. Classical

TABLE 3. Storage overhead of representative classical and quantum error-correction codes.

Domain	Code (params)	Phys : Log	Storage Overhead
Classical	Parity $(9, 8)$ (One parity bit per eight data bits [1])	1.125 : 1	12.5 %
Classical	SEC-DED Hamming/Hsiao $(72, 64)$ (Eight check bits per 64 data bits) [1].	1.125 : 1	12.5 %
Classical	ChipKill BCH/RS $(144, 128)$ [117]	1.125 : 1	12.5 %
Quantum	Shor [9,1,3] [14]	9 : 1	800 %
Quantum	Steane [7,1,3] [76]	7 : 1	600 %
Quantum	Surface code (distance 5) [114]	49 : 1	4 800 %
Quantum	Surface code (distance 7) [80]	101 : 1	10 000 %

Phys / Log = physical bits / qubits per protected logical bit / qubit. Storage overhead $= ((\text{Phys/Log}) - 1) \times 100\,\%$. Surface-code rows assume rotated planar patches: $n_{\text{phys}} = 2d^2 - 1$ (49 for d=5; 97 for d=7) plus 4 leakage-removal qubits on Willow, giving 101.

systems focus on reducing latency and improving energy efficiency across hierarchical memory layers, while quantum systems are limited by decoherence, error correction overhead, and cryogenic requirements. Both systems deal with scalability issues and high power consumption in large-scale applications [118].

1) LATENCY

In classical architectures, reducing memory latency is a key optimization challenge, as data access times between hierarchical memory levels (such as cache, random access memory (RAM), and secondary storage) have a major effect on system performance [3]. Techniques such as pipelining and prefetching are used at the hardware level to predict memory requests, reduce waiting time, and improve processing speed. Memory fragmentation at the operating system level can also increase latency and reduce how efficiently memory is used [68].

Quantum systems face different challenges. Latency in quantum memory depends on how long qubits can stay in a stable state (coherence time), how fast quantum gates can operate, and how well entanglement is maintained. Traditional methods like pipelining or speculative execution cannot be applied. Accessing or changing quantum memory requires precise quantum gate operations, and using Quantum Error Correction adds extra steps like syndrome measurements, which increase access time. This creates a trade-off: trying to access data faster can make qubits lose their state more quickly due to decoherence [119]. Figure 5 compares classical memory latencies with typical quantum-operation latencies, using a hard-disk seek as the slow-end reference point.

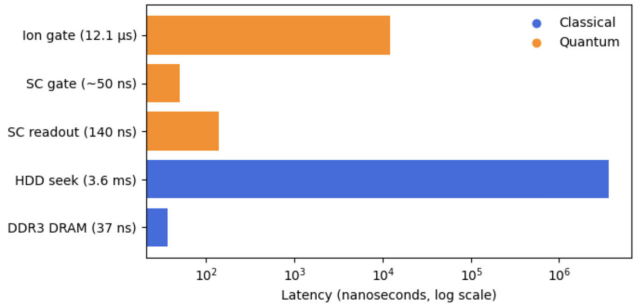

FIGURE 5. Latency range on a logarithmic scale (values from Table 4). Classical memory baseline: DDR3 DRAM read hit, 37 ns. Classical storage baseline: hard-disk seek, 3.6 ms (included only to show the full range). Quantum operations: 50–140 ns for superconducting qubits and 12.1 μs for a trapped-ion gate.

2) ENERGY CONSUMPTION

High energy consumption remains a significant challenge for both classical and quantum systems. While classical memory architectures use low-power DRAM, non-volatile memory, and optimized caching techniques to reduce energy consumption, large-scale computing environments still face high power demands, particularly in high-power computing and cloud computing infrastructures [28].

Quantum memory, however, introduces substantially higher energy demands due to its reliance on cryogenic cooling systems. Superconducting qubits require dilution refrigerators operating at millikelvin temperatures to maintain coherence, resulting in extreme energy consumption [116]. In a recent cryogenic-memory prototype that integrates spin-Hall-effect magnetic-tunnel-junction cells with superconducting hTron selectors, authors reported a write energy of $\sim$ 8 pJ bit^{-1} at 4K – before the dilution refrigerator's overhead factor is included [120]. In [121], authors give a 4.5 K specific-power of 5.77×10^2 W W^{-1}. Applying this overhead brings 8 pJ write to $\sim$ 4.6 nJ per bit. Furthermore, Quantum Error Correction techniques, such as surface codes and stabilizer codes, introduce additional energy overhead, as multiple physical qubits must be maintained to encode a single logical qubit. This overhead scales exponentially with increasing system size, making power efficiency a major challenge for practical quantum computing.

Alternative quantum memory technologies, such as trapped-ion qubits and photonic quantum memory operate at higher temperatures or remove the need for cryogenic cooling. However, trapped-ion systems require continuous high-power lasers for qubit control and cooling, along with ultra-high vacuum infrastructure. In [122], authors mentioned using two 5 mW Raman beams for a 50 μs gate, corresponding to an energy cost of approximately 0.5 μJ per trapped-ion entangling (logical) access. The significant gate latency for trapped ions is also a limitation. Photonic quantum memory relies on high-precision optical components, which impacts overall power efficiency. Figure 6 shows the per-access energy of state-of-the-art classical DRAM with two representative quantum/cryogenic technologies. Even

after refrigeration overhead, the cryo-spin write (4.6 nJ) is 100 times higher than Low Power Double Data Rate 2 (LPDDR2), and a trapped-ion gate (0.5 μJ) is a further two orders of magnitude above that – showing quantum memory is still far from classical efficiency.

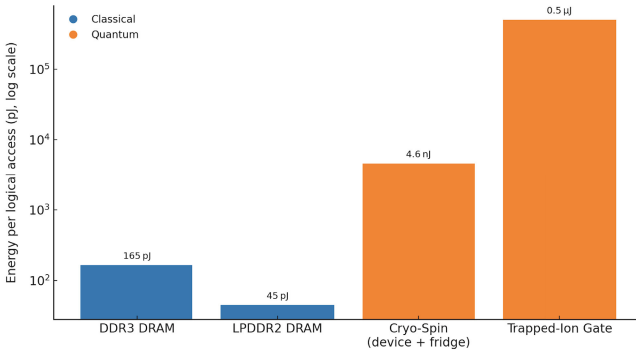

FIGURE 6. Per-access energy: DDR3, LPDDR2, cryo-spin (4.6 nJ), and trapped-ion (0.5 μJ); quantum energy requirements are orders of magnitude higher.

3) SCALABILITY

Classical memory systems scale efficiently using parallel memory channels, distributed computing architectures, and hierarchical caching mechanisms. These approaches help minimize memory bandwidth bottlenecks and maintain performance as workloads increase [68]. At the architectural level, advanced memory strategies such as Non-Uniform Memory Access-aware designs and Compute Express Link memory pooling improve resource sharing and performance in large-scale data centers.

Quantum memory, on the other hand, faces fundamental scalability challenges at the hardware level. The main issue is the requirement for fault-tolerant architectures, as qubit decoherence and gate errors make it difficult to scale without significant overhead from Quantum Error Correction [110]. Figure 7 shows representative raw error rates for DRAM and modern superconducting and trapped-ion qubits, highlighting the orders-of-magnitude gap that must be closed before quantum systems can match classical reliability. Scaling quantum memory involves increasing the number of reliable logical qubits while minimizing the number of required physical qubits, which is still a major research challenge (see Table 3). Achieving this requires improvements in hardware stability, inter-qubit connectivity, and coherence times. Without these advancements, building large-scale fault-tolerant quantum memory remains impractical. Trapped-ion memory offers the highest coherence and gate fidelity but faces slow gates and significant energy overhead from continuous-wave lasers. Superconducting qubits achieve low-latency gates, but their sub-kelvin cryogenic hardware and dense control wiring brings considerable power and scaling challenges.

Addressing these challenges is crucial for advancing memory management in both the classical and quantum computing paradigms. While classical systems continue to refine latency reduction techniques and energy-efficient

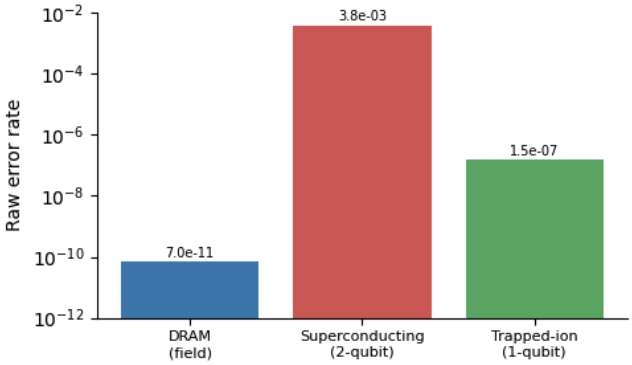

FIGURE 7. Raw physical error rates for a classical DRAM bit (field study) and two representative quantum operations (superconducting two-qubit fSim gate and trapped-ion single-qubit Clifford gate). DRAM values measure storage-time reliability, whereas qubit values are gate-operation fidelities.

architectures, quantum systems require fundamental innovations in hardware stability, fault tolerance, and Quantum Error Correction efficiency to achieve practical scalability. Table 4 presents a quantitative comparison of key metrics – latency, energy, error rate, and coherence – across classical and quantum memory systems. These parameters are selected to reflect core performance and reliability factors. While direct comparison is challenging due to the evolving nature of quantum hardware, the table illustrates fundamental gaps and emerging capabilities as quantum memory technologies advance. A broader qualitative analysis of memory management characteristics – including memory units, allocation strategies, access mechanisms, and scalability – is provided in Table 5.

TABLE 4. Quantitative comparison between classical and quantum memory systems.

Metric	Classical Memory	Quantum Memory
Latency	DRAM latency (37 ns - 2010); 3.6 ms access latency (Hard disk, 2010 Seagate) [1]	140 ns readout latency (Superconducting qubit [92]); 23–70 ns gate execution time (multi-qubit superconducting gates [95]); 12.1 μs gate-level latency for a $\pi/2$ microwave pulse (trapped-ion [100])
Energy per Access	70-260 pJ/bit (Double Data Rate type 3 (DDR3)); 40-50 pJ/bit (LPDDR2) [123]	∼8 pJ/bit write energy at 4 K for cryogenic quantum memory [120] (higher system-level energy due to refrigeration); Trapped-ion Raman gate: 0.5 µJ. [122]
Error Rate	2.5×10^{-11} to 7.0×10^{-11} failures bit^{-1} h^{-1} (25 000-70 000 FIT Mbit^{-1}, field study [111])	3.8×10^{-3} per two-qubit fSim gate (superconducting) [112]; 1.5×10^{-7} per single-qubit Clifford gate (trapped-ion $^{43}Ca^{+}$) [113]
Memory Coherence Time	Not applicable (data remains stable until overwritten)	∼50 s (Trapped-ion [100]); ∼500 μs (Photonic [98]); ∼60 ms (Silicon-based spin [103])

C. INTER-DEPENDENCIES IN HYBRID QUANTUM AND CLASSICAL SYSTEMS

Hybrid quantum-classical computation depend on efficient memory management to handle iterative optimizations,

TABLE 5. Comparative analysis of memory management in classical vs quantum architectures.

Aspect	Classical Memory Management	Quantum Memory Management
Memory Unit	Uses bits (0 or 1), which are the fundamental units of data in digital computation [1].	Uses qubits, which can exist in multiple states simultaneously (superposition), enabling quantum parallelism and complex data encoding [24].
Memory Structure	Organized in hierarchical levels (registers, cache, RAM, and secondary storage) to balance speed, capacity, and cost [28].	Uses quantum registers to store qubits, enabling quantum operations such as superposition and entanglement [48].
Memory Allocation Strategy	Uses static and dynamic memory allocation. Techniques such as paging, segmentation, and memory pooling optimize resource management [29], [33].	Depends on dynamic qubit allocation based on circuit design. Quantum memory models like Quantum Random Access Memory and error-correcting codes are required for stable qubit storage [59], [62].
Data Integrity	Data integrity is preserved through redundancy, ensuring reliable storage. Classical storage devices (e.g., RAM, SSD, HDD) degrade over time but are less susceptible to immediate data loss [28].	Quantum coherence is critical for maintaining data integrity. Qubits are highly sensitive to decoherence, leading to rapid loss of quantum information unless error correction techniques are applied [13], [22].
Memory Access	Memory access is optimized using caching, prefetching, and pipelining techniques, achieving low-latency [2].	Memory access is performed by executing controlled quantum gate operations that manipulate qubits stored in quantum registers [19].
Memory Retrieval	Classical systems have predictable, deterministic data retrieval with well-defined addressing schemes (e.g., virtual memory, paging, segmentation) [68].	Quantum memory retrieval involves state collapse upon measurement, resulting in a classical outcome while qubit loses its superposition [124]. Readout fidelity is a key challenge in quantum computing [18].
Energy Efficiency	Uses low-power DRAM, SRAM, and non-volatile memory (NVM) with optimization techniques such as dynamic voltage scaling (DVS) to reduce power consumption [5], [21]. Energy-efficient caching strategies further minimize computational overhead [31].	Requires cryogenic cooling (millikelvin temperatures) for superconducting qubits, leading to extreme energy consumption [49]. Trapped-ion qubits require continuous high-power laser control and ultra-high vacuum systems, while photonic qubits face photon loss and complex optical infrastructure challenges [53].
Scalability	Classical memory scales efficiently through hierarchical architectures, distributed systems, and hybrid storage solutions (e.g., CXL memory pooling) [34].	Quantum memory systems face significant scalability challenges due to high qubit error rates, decoherence, and hardware limitations. Large-scale fault-tolerant quantum systems require advanced QEC codes, which increase physical qubit overhead [79], [116].

store intermediate quantum states, and manage classical data preprocessing. Since quantum memory is limited by coherence time, classical systems have a crucial role in preprocessing data before quantum execution. Classical preprocessing, such as amplitude encoding, maps data to quantum states, while quantum operations such as the SWAP test evaluate similarities or fidelities between states [125]. Hybrid systems integrate classical and quantum resources to solve complex problems faced by classical computing. For example, Variational Quantum Algorithms highlight this interdependence, where quantum circuits compute expectation values for parameterized Hamiltonians, and classical optimizers iteratively update the parameters.

Task division is a fundamental aspect of hybrid quantum-classical computation. This can be observed in the Hamiltonian decomposition into Pauli strings where quantum processors compute the expectation values of individual Pauli terms, while classical systems aggregate these values and optimize the final results. Additionally, error mitigation highlights the collaboration between quantum and classical resources, where classical techniques refine noisy quantum outputs to enhance computational accuracy. These hybrid approaches balance the constraints of limited quantum resources with the computational efficiency of classical methods, leading to significant advancements in optimization, machine learning, and data analysis.

Interdependencies can be categorized as vertical and horizontal. Vertical interdependencies involve application-agnostic tasks such as decomposition, implementation, and control of quantum operations [126]. Horizontal interdependencies emphasize algorithmic workflows, including preprocessing, post-processing, and iterative quantum-classical interactions in variational algorithms [127], as well as problem decomposition into smaller tasks for quantum execution [128]. These interdependencies improve scalability, optimize quantum resource utilization, and support the seamless integration of hybrid quantum-classical computing.

However, these interdependencies present unique challenges. Frequent communication between quantum and classical systems introduces latency, particularly in iterative workflows such as Variational Quantum Eigensolver (VQE) and Quantum Approximate Optimization Algorithm (QAOA). Error propagation from noisy quantum outputs to classical post-processing can impact overall reliability, requiring robust error mitigation techniques like zero-noise extrapolation. Additionally, limitations in resources bring challenges in scalability, affecting the efficiency of both quantum and classical components as workloads increase.

To address inefficiencies caused by frequent quantum-classical communication, runtimes like Qiskit Runtime co-locate quantum and classical tasks, reducing latency and optimizing execution. Other platforms, like Amazon

Braket Hybrid Jobs and D-Wave Leap, also streamline task orchestration by integrating specialized tools for hybrid workflows. Such innovations enhance the management of interdependencies, enabling the development of efficient hybrid algorithms [129].

Quantum-classical interdependencies impact memory management, as hybrid systems must optimize data transfer between quantum and classical resources. Since quantum memory is volatile, intermediate quantum states must either be preserved in classical memory or recomputed iteratively. Optimizing these memory exchanges minimizes computational overhead and ensures efficient hybrid execution.

VI. EMERGING TRENDS AND FUTURE DIRECTIONS

Quantum computing is currently in the early stages, often referred as the Noisy Intermediate-Scale Quantum (NISQ) era. While NISQ devices hold significant potential, they also face challenges such as quantum noise and decoherence that are caused by interaction of quantum systems with their environment [130], [131]. When a quantum system S interacts with an environment E, the combined state changes into an entangled state [132]:

$$|\Psi\rangle = \sum_i c_i |s_i\rangle |E_i\rangle, \tag{5}$$

where $|s_i\rangle$ represents the system states, $|E_i\rangle$ denotes the environmental states, and c_i are corresponding probability amplitudes. By tracing out the environment, we can obtain the reduced density matrix for the system, $\rho_S = \mathrm{Tr}_E(\rho_{SE})$. This formulation shows how environmental interactions suppress the off-diagonal elements of the density matrix, which results in loss of quantum coherence. Over time, this decoherence follows an exponential decay, given by:

$$\langle E_i | E_j \rangle \propto e^{-t/\tau_d}, \tag{6}$$

where $\langle E_i | E_j \rangle$ represents the overlap between two environmental states, t is the time, and τ_d represents the decoherence time. Quantum memory stability is directly affected by decoherence, as qubits gradually lose their stored quantum information because of environmental interactions.

Quantum noise disrupts the stability of quantum memory by altering stored qubit states, leading to decoherence and information loss. The updated density matrix of quantum noise can be modeled using the Kraus representation [133], which is expressed as:

$$\rho' = \sum_\mu M_\mu \rho M_\mu^\dagger, \tag{7}$$

where M_μ are the Kraus operators satisfying the completeness condition: $\sum_\mu M_\mu^\dagger M_\mu = I$. This equation shows how noise transforms the state ρ, reducing both its coherence and fidelity. Consequently, preserving quantum states for reliable computation remains a fundamental challenge in quantum memory management.

Addressing quantum noise and decoherence is essential for achieving reliable quantum memory management and unlocking the full potential of quantum computing. Completely eliminating noise errors remains a significant challenge. While Quantum Error Correction techniques do not correct 100% of the errors, they help reduce the effect of noise. In fact, few-qubit codes such as the Steane code have demonstrated logical error rates as low as 10^{-15} under highly biased error channels, comparable to the reliability of classical memory systems [134]. However, when multiple errors occur in an uncontrolled manner, Quantum Error Correction can become ineffective, resulting in computation failure. The main goal of Fault-Tolerant Quantum Computers (FTQC) is to limit the propagation of faults within quantum circuits to maintain stable quantum memory [135]. Hardware advancements, including improved qubit coherence times, enhanced qubit connectivity, and more efficient quantum gates, are crucial to develop error-tolerant quantum memory architectures. However, current NISQ processors still lack full fault tolerance. Their limited qubit count, connectivity, and coherence times limit them to low-depth circuits. Researchers are actively working on error mitigation strategies, such as noise-aware compilation [136], zero-noise extrapolation [137], and dynamical decoupling [138], to extend qubit coherence and improve memory stability in near-term quantum systems. Additionally, Quantum Random Access Memory models are being explored to enhance data retrieval efficiency and reduce qubit overhead. Researchers are actively optimizing NISQ algorithms to achieve quantum advantages and solve certain problems faster than classical computers in the near future [139]. Hybrid quantum-classical approaches are emerging as practical solutions to solve the gaps between current limitations and practical applications.

A. CLASSICAL-QUANTUM DATA CONVERSION

A critical step in hybrid quantum-classical systems is the conversion of traditional binary data into quantum states that can be processed by qubits. Since most real-world information is stored classically, efficient encoding methods are essential for practical integration. Classical data can be represented as quantum states in the Hilbert space with a quantum feature map. This is known as quantum embedding, where a datapoint x is encoded as parameters for a quantum circuit which prepares the corresponding quantum state $|\psi_x\rangle$. We explore some embedding techniques below:

- Basis Encoding: Each bit string is mapped directly to a computational basis state, e.g., $x \in \{0, 1\}^n \mapsto |x\rangle$.
- Amplitude Encoding: A normalized classical N-dimensional datapoint x is represented by the amplitude of a n-qubit quantum state $|\psi_x\rangle$ as:

$$|\psi_x\rangle = \sum_{i=0}^{N-1} x_i |i\rangle. \tag{8}$$

- Rotation Encoding: Classical features are encoded into qubit rotation angles, e.g., $x \mapsto R_y(x)|0\rangle$.

These encoding schemes enable hybrid workflows where classical datasets are injected into quantum circuits for processing.

Just as classical data must be encoded into qubits, the results of quantum computation must be converted back into a classical representation for further processing. In practice, quantum-to-classical conversion is implemented in two main ways:

- Projective Measurement: Collapses each qubit into $|0\rangle$ or $|1\rangle$, producing a bitstring for classical storage. According to the Born rule [140], a single-qubit state

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad |\alpha|^2 + |\beta|^2 = 1 \tag{9}$$

  yields measurement probabilities

$$P(0) = |\alpha|^2, \quad P(1) = |\beta|^2. \tag{10}$$

  This means that upon measurement in the computational basis, the quantum state collapses to either $|0\rangle$ or $|1\rangle$, and the corresponding classical bit (0 or 1) is recorded.
- Expectation Estimation: Repeated measurements are used to estimate expectation values of observables [141], e.g.,

$$\langle O\rangle = \sum_x p(x)\, O(x), \tag{11}$$

  which are then stored in classical memory for use in optimization loops.

As shown in Figure 8, this pipeline for $N = 2$ demonstrates how classical data is embedded as qubit states and recovered via measurement.

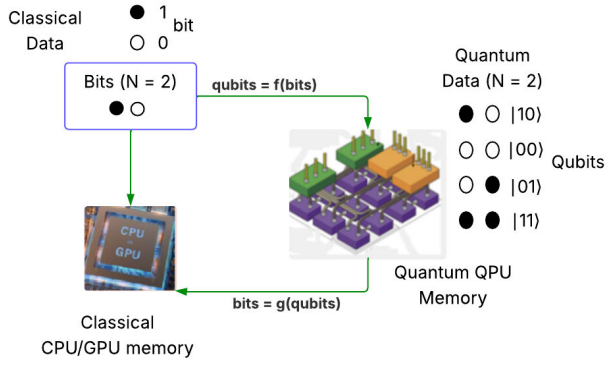

FIGURE 8. Conversion pipeline for $N = 2$: classical bits are embedded into quantum basis states via an encoding function f, while measurement/decoding function g maps quantum states back to classical bits. Quantum data spans all computational basis states, e.g., $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$.

B. HYBRID QUANTUM-CLASSICAL SYSTEMS

While some quantum algorithms do not rely on Quantum Random Access Memory, they still illustrate important patterns of memory coordination in hybrid quantum-classical systems. In particular, hybrid algorithms such as the Variational Quantum Eigensolver and the Quantum Approximate Optimization Algorithm illustrate how memory responsibilities are distributed between quantum and classical components. Classical systems handle storage-intensive tasks – such as parameter optimization and data encoding – while quantum systems store small, low-depth circuits limited by coherence time. Similarly, Quantum Kernel Estimation (QKE) and hybrid Quantum Generative Adversarial Networks (QGANs) demonstrate how quantum subroutines can be integrated into classical machine learning workflows, with quantum registers used transiently for feature mapping or data generation while classical systems manage large-scale optimization and storage. Although these algorithms do not access large quantum memory structures, they reflect emerging trends in hybrid memory management. Memory efficiency is achieved by combining classical storage with quantum registers. Their analysis helps to clarify which classical strategies translate into useful patterns for quantum workflows and where new paradigms are needed.

1) VARIATIONAL QUANTUM EIGENSOLVER

The Variational Quantum Eigensolver optimizes memory management in hybrid quantum-classical computing by reducing quantum memory requirements through parameterized ansatz and classical optimization. Instead of storing full wavefunctions, VQE approximates eigenstates variationally, minimizing the expectation value of the Hamiltonian [141]:

$$\langle H\rangle(q) = \langle \Psi(q)|H|\Psi(q)\rangle, \tag{12}$$

where $\Psi(q)$ is the parameterized trial state and $\langle H\rangle(q)$ is the expectation value to be minimized. The Hamiltonian decomposition into a sum of Pauli terms:

$$H = \sum_\alpha h_\alpha O_\alpha, \tag{13}$$

where h_α are the calculated coefficients and O_α are the tensor product of Pauli matrices. This allows independent measurements, while operator grouping reduces the number of measurements by exploiting commutativity. The unitary coupled cluster (UCC) ansatz is given as:

$$|\Psi_{CC}^{(k)}(\theta)\rangle = \exp(T^{(k)}(\theta))|\Phi_R\rangle, \tag{14}$$

where Φ_R is the reference state and $\exp(T^{(k)}(\theta))$ is the cluster excitation operator. It stores only essential variational parameters, minimizing memory overhead. Variational error suppression further enhances efficiency by reducing errors without requiring extensive error correction. Additionally, adaptive measurement strategies focus on dominant terms, reducing redundant storage.

Although VQE does not utilize Quantum Random Access Memory, it remains a relevant example of memory-aware quantum algorithm design. The quantum circuit is kept shallow and narrow, while classical systems carry out memory-intensive optimization and parameter updates. This division of responsibility highlights an emerging pattern in hybrid systems, where efficient memory management is achieved through the complementary roles of classical and quantum components. Figure 9 illustrates how quantum

(state preparation and measurement) and classical (parameter optimization) components collaborate to reduce quantum memory overhead in VQE.

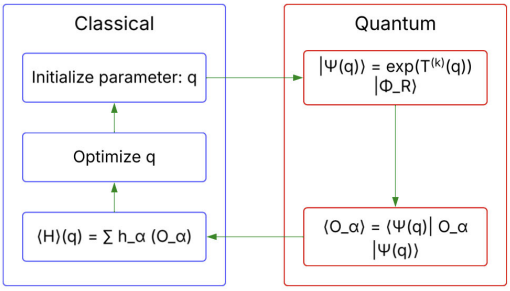

FIGURE 9. Hybrid quantum-classical workflow in a variational algorithm. A classical optimizer updates the parameters q, which define a parameterized quantum state $\Psi(q)$. The quantum processor estimates the expectation values $\langle \mathcal{O}_\alpha \rangle$, and the classical system computes the total energy $\langle H \rangle(q)$, feeding back into the parameter update loop.

2) QUANTUM APPROXIMATE OPTIMIZATION ALGORITHM

The Quantum Approximate Optimization Algorithm is a hybrid quantum-classical algorithm for solving combinatorial optimization problems by producing approximate solutions. QAOA operates by alternating between the application of two parameterized quantum operators: the cost Hamiltonian H_C, which encodes the optimization problem, and the mixer Hamiltonian H_B, which ensures exploration of the solution space.

Starting with a uniform superposition state prepared using Hadamard gates, the algorithm iteratively applies unitary transformations to the p layers, producing a final quantum state $|\gamma, \beta\rangle$. The parameters (γ, β) are classically optimized to minimize the expectation value of H_C [142], defined as:

$$F_p(\gamma, \beta) = \langle \psi(\gamma, \beta) | H_C | \psi(\gamma, \beta) \rangle \tag{15}$$

thereby approximating the optimal solution. Memory management in QAOA is influenced by the structure of the variational circuit, where quantum states are stored in quantum registers throughout the optimization process. The algorithm maintains state information across multiple layers by applying unitary operators, with qubit rotations representing cost and mixer transformations. As the circuit depth p increases, memory overhead grows, and more quantum resources are required to maintain coherence. The classical memory is used to iteratively update the parameters (γ, β) optimizing them based on measurement results obtained from quantum state collapses. Due to the computational requirements of repeated quantum-classical feedback loops, efficient data handling and parameter storage play a crucial role in ensuring the scalability of QAOA (see Figure 10).

QAOA demonstrates a memory-efficient hybrid design where the quantum circuit encodes only a fixed-depth variational state, while classical memory handles parameter updates and result aggregation. As the circuit depth increases, maintaining coherence becomes more demanding, but the overall memory burden remains primarily on the classical side. This division enables scalability by offloading storage and optimization to classical processors, while keeping quantum memory usage minimal and focused.

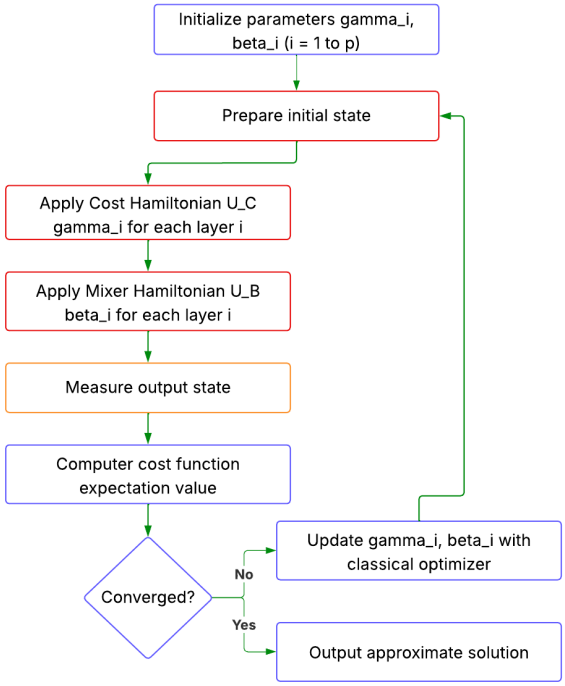

FIGURE 10. Conceptual workflow of QAOA, illustrating the iterative application of the cost Hamiltonian $U_C(\gamma_i) = e^{-i\gamma_i H_C}$ and mixer Hamiltonian $U_B(\beta_i) = e^{-i\beta_i H_B}$ across p layers, with classical optimization of parameters (γ, β) to approximate the optimal solution.

3) QUANTUM KERNEL ESTIMATION

Quantum Kernel Estimation constructs the hyperplane using a classical SVM while using the quantum computer to estimate the kernel function [143]. The quantum computer is used twice in this protocol. In the training phase, the kernel

$$K(x, z) = |\langle \phi(x) \mid \phi(z) \rangle|^2 \tag{16}$$

is estimated on a quantum computer for all pairs of training samples $x, z \in T$. The kernel is then used in the Wolfe–dual SVM to find the optimal hyperplane. In the classification phase, the quantum computer is used again to estimate $K(s, x)$ for a new datum s and the support vectors $x \in T$ obtained from the optimization. This is sufficient to construct the full SVM classifier.

To estimate the inner product for the kernel, the circuit $U_\phi(x)^\dagger U_\phi(z)$ is applied to $|0^n\rangle$, as illustrated in the hybrid workflow of Figure 11. The final state is measured in the Z-basis R times and the frequency of observing the $|0^n\rangle$ string is taken as the transition probability. Each kernel entry is obtained to an additive sampling error of $O(R^{-1/2})$. An estimator $\hat{K}$ that deviates from the exact kernel K by at most δ in operator norm can be obtained with

$$R = O(\delta^{-2}|T|^4) \tag{17}$$

shots in total.

This hybrid division highlights that quantum registers are used only transiently to estimate pairwise overlaps, while the large-scale storage and optimization of the kernel matrix is handled by classical memory. A practical challenge in QKE is that the feature map circuits must remain shallow to mitigate noise on NISQ devices, which restricts expressiveness. The classical kernel storage still scales quadratically with the training set size.

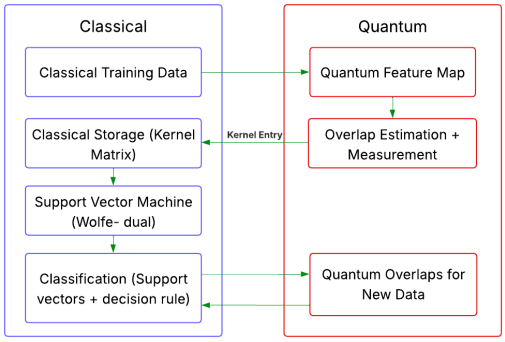

FIGURE 11. Hybrid workflow of Quantum Kernel Estimation. The quantum computer is responsible for generating feature maps and estimating overlaps through measurements, while the classical side stores the kernel matrix, trains the SVM via the Wolfe-dual formulation, and performs classification using support vectors. Quantum resources are called again in the classification phase to estimate overlaps for new data points.

4) HYBRID QUANTUM GENERATIVE ADVERSARIAL NETWORK

Hybrid quantum–classical GANs illustrate another pattern of memory coordination in NISQ-era algorithms. In this approach, a quantum circuit serves as the generator while a classical neural network functions as the discriminator [144]. The generator is implemented as a parameterized quantum circuit (PQC), which maps a latent vector z into a quantum state. The discriminator evaluates real and generated samples using the Wasserstein GAN with gradient penalty (WGAN-GP) loss:

$$\min_G \max_{D\in\mathcal{D}} \mathbb{E}_{x\sim P_{\text{data}}}[D(x)] - \mathbb{E}_{z\sim P_z}[D(G(z))] - \lambda\, \mathbb{E}_{\hat{x}\sim P_{\hat{x}}} \left(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1\right)^2, \quad (18)$$

where G denotes the quantum generator, D is the classical discriminator, and λ is a regularization constant. The quantum generator is trained via parameter-shift rules, while the discriminator backpropagates using classical GPU-based optimization.

This division of responsibilities reflects a hybrid memory strategy: quantum registers are used transiently to generate data patches from latent variables, while the classical system stores training batches, discriminator weights, and optimizer states. Such partitioning reduces quantum memory overhead but introduces challenges in synchronization between quantum-generated data and classical batch processing (see Figure 12).

Hybrid quantum-classical models integrate quantum speedups with classical optimization to enhance computational efficiency while minimizing quantum memory demands. Algorithms such as VQE and QAOA illustrate a memory-aware design pattern in which quantum hardware executes shallow, parameterized circuits, while classical systems handle memory-intensive optimization and data aggregation. Similarly, Hybrid Quantum Neural Networks and Quantum Support Vector Machines use quantum circuits for complex feature extraction or kernel embedding, offloading model training and inference to classical processors with robust memory infrastructures. These examples reflect an emerging principle in hybrid system design: quantum resources are applied selectively for their computational advantages, while classical systems maintain responsibility for scale, stability, and memory handling. This mirrors the limitations of current quantum memory technologies, including QRAM, and emphasizes the need for practical coordination strategies between quantum registers and classical memory layers. By depending on low-depth circuits and shifting memory-intensive operations to classical systems, these hybrid approaches are well-suited to the limitations of current NISQ-era hardware.

A key challenge in hybrid systems is the coordination between classical and quantum memory. Data must frequently cross the quantum-classical boundary: parameters, encoded inputs, and measurement results are transferred back and forth which can create bottlenecks. Synchronization is critical in these systems as deterministic classical operations must align properly with probabilistic quantum measurements. Errors from quantum registers can propagate into classical post-processing if not detected and mitigated early. Addressing these coordination challenges is essential for scalable hybrid memory management.

C. FUTURE RESEARCH DIRECTIONS

One of the key research directions in memory management is the development of hybrid quantum-classical systems that optimize data exchange across the quantum-classical boundary. Currently, quantum computation relies heavily on classical resources for pre-processing, post-processing, and parameter optimization, making seamless integration essential. Future work must explore efficient allocation strategies that reduce communication overhead, balance quantum register usage, and mitigate synchronization bottlenecks that arise when measurement results are repeatedly transferred between devices. Hybrid algorithms already show promise in machine learning, optimization, and generative modeling, but their scalability depends on memory-aware designs. Research on hybrid quantum-classical memory models could help overcome data bottlenecks in near-term processors by exploiting quantum-assisted cache management, quantum-enhanced compression, and adaptive allocation schemes. Beyond algorithmic improvements, domain-specific integration of hybrid memory systems in areas such as cybersecurity,

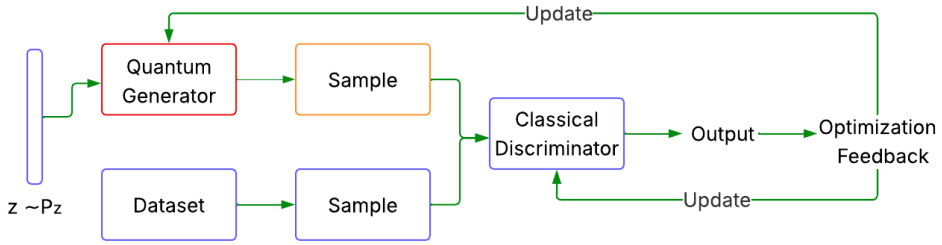

FIGURE 12. Hybrid Quantum-Classical GAN: a quantum generator (PQC) produces samples from latent inputs, while a classical discriminator evaluates real vs. generated data and provides optimization feedback.

finance, healthcare, and drug discovery will be crucial for achieving practical advantages [131].

Another crucial area is Quantum Error Correction and fault tolerance, as quantum systems are highly susceptible to decoherence and noise. Recent research has increasingly focused on developing low-overhead error correction techniques, particularly using quantum low-density parity-check codes such as Bivariate Bicycle codes and Quantum Tanner codes. These alternatives offer higher encoding rates, linear distance scaling, and reduced physical qubit overhead compared to traditional codes. Additionally, machine learning-driven error correction is becoming popular, which provides adaptive noise suppression and syndrome decoding methods to enhance resilience against quantum errors.

Optimizing memory access and retrieval in quantum systems is another critical area for research. Unlike classical memory, quantum memory lacks direct addressing mechanisms and depends on quantum gate execution for data access, which introduces latency. Researchers need to design fast quantum read/write mechanisms that improve the reliability of qubit storage and retrieval. Optimizing quantum gate execution times and exploring novel qubit addressing schemes is essential for improving efficiency in Quantum Random Access Memory and other emerging quantum architectures.

Scalability remains one of the biggest challenges in quantum computing. Future research should focus on scalable quantum memory architectures that enhance qubit connectivity and information stability. Modular quantum memory systems and distributed quantum memory networks, leveraging entanglement for long-distance quantum communication, could provide new directions for scalability. Additionally, developing fault-tolerant memory models that minimize the number of redundant qubits while maintaining computational accuracy is crucial to advance quantum technology.

Another research direction is energy-efficient quantum memory solutions. Quantum computing requires cryogenic cooling, particularly for superconducting qubits, leading to substantial energy consumption. Investigating alternative quantum memory hardware could reduce energy demands. However, limitations such as high-power laser requirements, significant gate latency for trapped ions and photon loss in photonic qubits needs to be addressed. Additionally, optimizing quantum error mitigation techniques that lower the energy overhead associated with QEC will be crucial to make quantum computing more sustainable.

Summary of Research Questions' Answers: From the analyses presented across Sections III–VI, we can now synthesize the main findings in response to our research questions. Classical memory systems achieve performance and scalability through hierarchical organization, deterministic access control, and efficient allocation strategies such as caching, paging, and virtual memory. Together, these strategies make classical memory fast, reliable, and well-suited for large-scale computing. On the other hand, quantum memory management is fundamentally different from these classical mechanisms. Quantum computing relies on qubits that require continuous stabilization and sophisticated Quantum Error Correction to preserve coherence against decoherence and noise. Quantum read/write operations strongly influence overall memory fidelity, as pulse optimization, initialization fidelity, and gate precision directly determine qubit coherence times across different qubit platforms. When we evaluate systems in terms of latency, energy efficiency, and scalability, classical systems benefit from decades of optimization, whereas quantum systems remain constrained by cryogenic energy demands, high gate latencies, and the heavy overhead of fault tolerance. Hybrid quantum-classical systems help us overcome some of these limitations by allocating optimization, parameter tuning, and data aggregation to classical processors while quantum resources handle low-depth computations. Finally, ongoing challenges and research opportunities center on reducing error-correction overhead, improving cryogenic efficiency, and designing scalable, energy-aware quantum memory architectures that integrate seamlessly with classical infrastructure. These findings collectively address all six research questions and demonstrate how classical and quantum memory management are evolving in complementary ways toward hybrid, fault-tolerant computing systems.

VII. CONCLUSION

In this study, we explored the differences in memory management between classical and quantum architectures. We examined various aspects including memory access and allocation, error correction techniques, and scalability challenges. We analyzed the classical memory hierarchy along with key techniques like virtual memory, caching,

and memory pooling, highlighting their role in improving system efficiency and scalability. In the quantum domain, we discussed quantum registers, quantum gates, QRAM, Quantum Error Correction methods, and quantum read/write operations. Additionally, we investigated hybrid quantum-classical systems, emphasizing their role in optimizing computational efficiency in the NISQ era.

Classical systems depend on well-structured hierarchical memory, deterministic access patterns, and robust error correction techniques to ensure scalability and efficiency. On the other hand, quantum systems must address significant challenges, including decoherence and noise issues. While classical memory management has been refined over decades, quantum memory management is still in its early stages, requiring novel approaches to improve stability and fault tolerance. Future advancements in hybrid classical-quantum memory integration, Quantum Error Correction and scalable quantum architectures will be crucial to overcome current limitations. Further research in energy-efficient quantum memory, dynamic memory allocation for quantum registers, and optimized quantum read/write mechanisms will enhance the practical applications of quantum computing. As quantum technologies continue to evolve, memory management will have a vital role in the transition from small-scale quantum processors to large-scale fault-tolerant quantum systems.

ACKNOWLEDGMENT

However, any opinions, findings, conclusions, or recommendations expressed in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the funding agencies.

REFERENCES

[1] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. Amsterdam, The Netherlands: Elsevier, 2011.

[2] W. A. Wulf and S. A. McKee, "Hitting the memory wall: Implications of the obvious," *ACM SIGARCH Comput. Archit. News*, vol. 23, no. 1, pp. 20–24, Mar. 1995.

[3] P. R. Wilson, "Uniprocessor garbage collection techniques," in *Proc. Int. Workshop Memory Manage.*, 2005, pp. 1–42.

[4] B. Maity, B. Donyanavard, and N. Dutt, "Self-aware memory management for emerging energy-efficient architectures," in *Proc. 11th Int. Green Sustain. Comput. Workshops (IGSC)*, Oct. 2020, pp. 1–8.

[5] H.-L. Li, C.-L. Yang, and H.-W. Tseng, "Energy-aware flash memory management in virtual memory system," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 16, no. 8, pp. 952–964, Aug. 2008.

[6] J. Zhang, S. Ho Yeung, Y. Shu, B. He, and W. Wang, "Efficient memory management for GPU-based deep learning systems," 2019, *arXiv:1903.06631*.

[7] L. Wang, J. Ye, Y. Zhao, W. Wu, A. Li, S. L. Song, Z. Xu, and T. Kraska, "Superneurons: Dynamic GPU memory management for training deep neural networks," in *Proc. ACM SIGPLAN Notices*, Mar. 2018, vol. 53, no. 1, pp. 41–53.

[8] X. Peng, X. Shi, H. Dai, H. Jin, W. Ma, Q. Xiong, F. Yang, and X. Qian, "Capuchin: Tensor-based GPU memory management for deep learning," in *Proc. 25th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Mar. 2020, pp. 891–905.

[9] S. Bateni, Z. Wang, Y. Zhu, Y. Hu, and C. Liu, "Co-optimizing performance and memory footprint via integrated CPU/GPU memory management, an implementation on autonomous driving platform," in *Proc. IEEE Real-Time Embedded Technol. Appl. Symp. (RTAS)*, Apr. 2020, pp. 310–323.

[10] P. Gangadhar, M. V. Rao, A. K. Hota, and V. V. Rao, "Distributed memory and cpu management in cloud computing environment," *Int. J. Appl. Eng. Res.*, vol. 12, no. 24, pp. 15972–15978, 2017.

[11] D. R. Patil, "Dynamic resource allocation and memory management using machine learning for cloud environments," *Int. J. Adv. Trends Comput. Sci. Eng.*, vol. 9, no. 4, pp. 5921–5927, Aug. 2020.

[12] A. Pupykina and G. Agosta, "Survey of memory management techniques for HPC and cloud computing," *IEEE Access*, vol. 7, pp. 167351–167373, 2019.

[13] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, Aug. 2018.

[14] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory," *Phys. Rev. A, Gen. Phys.*, vol. 52, no. 4, pp. R2493–R2496, Oct. 1995.

[15] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Phys. Rev. A, Gen. Phys.*, vol. 86, no. 3, Sep. 2012, Art. no. 032324.

[16] G. Meuli, M. Soeken, M. Roetteler, N. Bjorner, and G. D. Micheli, "Reversible pebbling game for quantum memory management," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2019, pp. 288–291.

[17] P. Promponas, V. Valls, S. Guha, and L. Tassiulas, "Maximizing entanglement rates via efficient memory management in flexible quantum switches," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 7, pp. 1749–1762, Jul. 2024.

[18] C. Liu, M. Wang, S. A. Stein, Y. Ding, and A. Li, "Quantum memory: A missing piece in quantum computing units," 2023, *arXiv:2309.14432*.

[19] W. Zhang, D.-S. Ding, Y.-B. Sheng, L. Zhou, B.-S. Shi, and G.-C. Guo, "Quantum secure direct communication with quantum memory," *Phys. Rev. Lett.*, vol. 118, no. 22, May 2017, Art. no. 220501.

[20] R. Palanivel and P. Muthulakshmi, "Design and analysis of parallel quantum transfer fractal priority replay with dynamic memory algorithm in quantum reinforcement learning for robotics," *IET Quantum Commun.*, vol. 5, no. 4, pp. 360–383, Dec. 2024.

[21] D. Brooks and M. Martonosi, "Dynamic thermal management for high-performance microprocessors," in *Proc. HPCA 7th Int. Symp. High-Perform. Comput. Archit.*, Jan. 2001, pp. 171–182.

[22] S. J. Devitt, W. J. Munro, and K. Nemoto, "Quantum error correction for beginners," *Rep. Prog. Phys.*, vol. 76, no. 7, Jul. 2013, Art. no. 076001.

[23] M. Schuld, I. Sinayskiy, and F. Petruccione, "An introduction to quantum machine learning," *Contemp. Phys.*, vol. 56, no. 2, pp. 172–185, 2014.

[24] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge, U.K.: Cambridge Univ. Press, 2010.

[25] F. J. Duarte, *Fundamentals Quantum Entanglement*. Bristol, U.K.: IOP Publishing, 2022.

[26] Y. Zhang, L. Li, Z. Lu, A. Jantsch, M. Gao, H. Pan, and F. Han, "A survey of memory architecture for 3D chip multi-processors," *Microprocessors Microsystems*, vol. 38, no. 5, pp. 415–430, Jul. 2014.

[27] R. Pandey and A. Sahu, "Performance and area trade-off of 3D-stacked DRAM based chip multiprocessor with hybrid interconnect," *IEEE Trans. Emerg. Topics Comput.*, vol. 9, no. 4, pp. 1945–1959, Oct. 2021.

[28] A. S. Tanenbaum and H. Bos, *Modern Operating Systems*. London, U.K.: Pearson Education, 2015.

[29] O. Kwon, Y. Lee, J. Park, S. Jang, B. Tak, and S. Hong, "Distributed page table: Harnessing physical memory as an unbounded hashed page table," in *Proc. 57th IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, May 2024, pp. 36–49.

[30] H. Torabi, H. Khazaei, and M. Litoiu, "A learning-based caching mechanism for edge content delivery," in *Proc. 15th ACM/SPEC Int. Conf. Perform. Eng.*, May 2024, pp. 236–246.

[31] S. Rahman, M. G. R. Alam, and M. M. Rahman, "Deep learning-based predictive caching in the edge of a network," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2020, pp. 797–801.

[32] M. T. Banday and M. Khan, "A study of recent advances in cache memories," in *Proc. Int. Conf. Contemp. Comput. Informat. (IC3I)*, Nov. 2014, pp. 398–403.

[33] N. V. Patil, "Dynamic memory allocation: Role in memory management," *Int. J. Current Eng. Technol.*, vol. 4, no. 2, pp. 531–535, 2014.

[34] D. Gouk, M. Kwon, H. Bae, S. Lee, and M. Jung, "Memory pooling with CXL," *IEEE Micro*, vol. 43, no. 2, pp. 48–57, Mar. 2023.

[35] A. Cho and A. Daglis, "StarNUMA: Mitigating NUMA challenges with memory pooling," in *Proc. 57th IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Nov. 2024, pp. 997–1012.

[36] B. Kenwright, "Fast efficient fixed-size memory pool: No loops and no overhead," 2022, *arXiv:2210.16471*.

[37] R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Trans. Device Mater. Rel.*, vol. 5, no. 3, pp. 305–316, Sep. 2005.

[38] S. Tam, "Single error correction and double error detection," *Xilinx Appl. Note*, vol. 645, pp. 1–12, Aug. 2006.

[39] W. Fitriani, A. Putera, and U. Siahaan, "Single-bit parity detection and correction using Hamming code 7-bit model," *Int. J. Comput. Appl.*, vol. 154, no. 2, pp. 12–16, Nov. 2016.

[40] M. Poolakkaparambil, J. Mathew, A. Jabir, and S. P. Mohanty, "Low complexity cross parity codes for multiple and random bit error correction," in *Proc. 13th Int. Symp. Quality Electron. Design (ISQED)*, Mar. 2012, pp. 57–62.

[41] I. Alam, C. Schoeny, L. Dolecek, and P. Gupta, "Parity++: Lightweight error correction for last level caches," in *Proc. 48th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. Workshops (DSN-W)*, Jun. 2018, pp. 114–120.

[42] S. Liu, P. Reviriego, and F. Lombardi, "Detection of limited magnitude errors in emerging multilevel cell memories by one-bit parity (OBP) or two-bit parity (TBP)," *IEEE Trans. Emerg. Topics Comput.*, vol. 9, no. 4, pp. 1792–1802, Oct. 2021.

[43] R. Medico, D. Spina, D. Vande Ginste, D. Deschrijver, and T. Dhaene, "Machine-learning-based error detection and design optimization in signal integrity applications," *IEEE Trans. Compon., Packag., Manuf. Technol.*, vol. 9, no. 9, pp. 1712–1720, Sep. 2019.

[44] R. Swischuk and D. Allaire, "A machine learning approach to aircraft sensor error detection and correction," *J. Comput. Inf. Sci. Eng.*, vol. 19, no. 4, Dec. 2019, Art. no. 041009.

[45] A. J. Zimolzak, L. Wei, U. Mir, A. Gupta, V. Vaghani, D. Subramanian, and H. Singh, "Machine learning to enhance electronic detection of diagnostic errors," *JAMA Netw. Open*, vol. 7, no. 9, Sep. 2024, Art. no. e2431982.

[46] K. Prall, "Benchmarking and metrics for emerging memory," in *Proc. IEEE Int. Memory Workshop (IMW)*, May 2017, pp. 1–5.

[47] P. M. Chen and D. A. Patterson, "Storage performance-metrics and benchmarks," *Proc. IEEE*, vol. 81, no. 8, pp. 1151–1165, 1993.

[48] P. I. Hagouel and I. G. Karafyllidis, "Quantum computers: Registers, gates and algorithms," in *28th Int. Conf. Microelectron. Proc.*, May 2012, pp. 15–21.

[49] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver, "A quantum engineer's guide to superconducting qubits," *Appl. Phys. Rev.*, vol. 6, no. 2, pp. 021318-1–021318-57, Jun. 2019.

[50] A. Chatterjee, K. Phalak, and S. Ghosh, "Quantum error correction for dummies," in *Proc. IEEE Int. Conf. Quantum Comput. Eng. (QCE)*, Sep. 2023, pp. 70–81.

[51] D. Schrader, I. Dotsenko, M. Khudaverdyan, Y. Miroshnychenko, A. Rauschenbeutel, and D. Meschede, "Neutral atom quantum register," *Phys. Rev. Lett.*, vol. 93, no. 15, Oct. 2004, Art. no. 150501.

[52] S. Slussarenko and G. J. Pryde, "Photonic quantum information processing: A concise review," *Appl. Phys. Rev.*, vol. 6, no. 4, pp. 041303-1–041303-19, Dec. 2019.

[53] C. D. Bruzewicz, J. Chiaverini, R. McConnell, and J. M. Sage, "Trapped-ion quantum computing: Progress and challenges," *Appl. Phys. Rev.*, vol. 6, no. 2, Jun. 2019, Art. no. 021314.

[54] G. Burkard, T. D. Ladd, A. Pan, J. M. Nichol, and J. R. Petta, "Semiconductor spin qubits," *Rev. Modern Phys.*, vol. 95, no. 2, Jun. 2023, Art. no. 025003.

[55] M. Freedman, A. Kitaev, M. Larsen, and Z. Wang, "Topological quantum computation," *Bull. Amer. Math. Soc.*, vol. 40, no. 1, pp. 31–38, 2002.

[56] J. Preskill, "Reliable quantum computers," *Proc. Roy. Soc. London. A, Math., Phys. Eng. Sci.*, vol. 454, no. 1969, pp. 385–410, Jan. 1998.

[57] G. E. Crooks, "Gates, states, and circuits," Tech. Rep., 2020. Accessed: Aug. 10, 2025. [Online]. Available: https://threeplusone.com/notes/

[58] A. Ekert, P. Hayden, and H. Inamori, "Basic concepts in quantum computation," in *Proc. Coherent At. Matter Waves*, Aug. 2000, pp. 661–701.

[59] C. T. Hann, G. Lee, S. M. Girvin, and L. Jiang, "Resilience of quantum random access memory to generic noise," *PRX Quantum*, vol. 2, no. 2, Apr. 2021, Art. no. 020311.

[60] S. Jaques and A. G. Rattew, "QRAM: A survey and critique," 2023, *arXiv:2305.10310*.

[61] P. Mukhopadhyay, "A quantum random access memory (QRAM) using a polynomial encoding of binary strings," *Sci. Rep.*, vol. 15, no. 1, p. 11002, Mar. 2025.

[62] K. Phalak, A. Chatterjee, and S. Ghosh, "Quantum random access memory for dummies," *Sensors*, vol. 23, no. 17, p. 7462, Aug. 2023.

[63] V. Giovannetti, S. Lloyd, and L. Maccone, "Architectures for a quantum random access memory," *Phys. Rev. A, Gen. Phys.*, vol. 78, no. 5, Nov. 2008, Art. no. 052310.

[64] R. Beals, S. Brierley, O. Gray, A. W. Harrow, S. Kutin, N. Linden, D. Shepherd, and M. Stather, "Efficient distributed quantum computing," *Proc. Roy. Soc. A, Math.*, vol. 469, no. 2153, 2013, Art. no. 20120686.

[65] D. K. Park, F. Petruccione, and J.-K.-K. Rhee, "Circuit-based quantum random access memory for classical data," *Sci. Rep.*, vol. 9, no. 1, p. 3949, Mar. 2019.

[66] J. M. Baker, C. Duckering, and F. T. Chong, "Efficient quantum circuit decompositions via intermediate qudits," in *Proc. IEEE 50th Int. Symp. Multiple-Valued Log. (ISMVL)*, Apr. 2020, pp. 303–308.

[67] M. Y. Niu, A. Zlokapa, M. Broughton, S. Boixo, M. Mohseni, V. Smelyanskyi, and H. Neven, "Entangling quantum generative adversarial networks," *Phys. Rev. Lett.*, vol. 128, no. 22, Jun. 2022, Art. no. 220505.

[68] K. Phalak, J. Li, and S. Ghosh, "Approximate quantum random access memory architectures," 2022, *arXiv:2210.14804*.

[69] S. Xu, A. Lu, and Y. Ding, "Fat-tree QRAM: A high-bandwidth shared quantum random access memory for parallel queries," 2025, *arXiv:2502.06767*.

[70] Y. Wang, Y. Alexeev, L. Jiang, F. T. Chong, and J. Liu, "Fundamental causal bounds of quantum random access memories," *npj Quantum Inf.*, vol. 10, no. 1, p. 71, Jul. 2024.

[71] B. D. Clader, A. M. Dalzell, N. Stamatopoulos, G. Salton, M. Berta, and W. J. Zeng, "Quantum resources required to block-encode a matrix of classical data," *IEEE Trans. Quantum Eng.*, vol. 3, pp. 1–23, 2022.

[72] D. S. Steiger and M. Troyer, "Racing in parallel: Quantum versus classical," *APS March Meeting Abstr.*, vol. 2016, pp. H44–010, 2016.

[73] A. R. Calderbank and P. W. Shor, "Good quantum error-correcting codes exist," *Phys. Rev. A, Gen. Phys.*, vol. 54, no. 2, pp. 1098–1105, Aug. 1996.

[74] P. W. Shor, "Fault-tolerant quantum computation," in *Proc. 37th Conf. Found. Comput. Sci.*, Oct. 1996, pp. 56–65.

[75] O. O. Khalifa, N. A. B. Sharif, R. A. Saeed, S. Abdel-Khalek, A. N. Alharbi, and A. A. Alkathiri, "Digital system design for quantum error correction codes," *Contrast Media Mol. Imag.*, vol. 2021, no. 1, pp. 1–8, 2021.

[76] A. M. Steane, "Simple quantum error-correcting codes," *Phys. Rev. A, Gen. Phys.*, vol. 54, no. 6, pp. 4741–4751, Dec. 1996.

[77] A. Y. Kitaev, "Fault-tolerant quantum computation by anyons," *Ann. Phys.*, vol. 303, no. 1, pp. 2–30, Jan. 2003.

[78] S. Bravyi, M. Englbrecht, R. König, and N. Peard, "Correcting coherent errors with surface codes," *npj Quantum Inf.*, vol. 4, no. 1, p. 55, Oct. 2018.

[79] S. Krinner, N. Lacroix, A. Remm, A. Di Paolo, E. Genois, C. Leroux, C. Hellings, S. Lazar, F. Swiadek, J. Herrmann, G. J. Norris, C. K. Andersen, M. Müller, A. Blais, C. Eichler, and A. Wallraff, "Realizing repeated quantum error correction in a distance-three surface code," *Nature*, vol. 605, no. 7911, pp. 669–674, May 2022.

[80] R. Acharya et al., "Quantum error correction below the surface code threshold," *Nature*, vol. 638, pp. 920–926, Dec. 2024.

[81] H. Bombin and M. A. Martin-Delgado, "Topological quantum distillation," *Phys. Rev. Lett.*, vol. 97, no. 18, Oct. 2006, Art. no. 180501.

[82] D. Bacon, "Operator quantum error-correcting subsystems for self-correcting quantum memories," *Phys. Rev. A, Gen. Phys.*, vol. 73, no. 1, Jan. 2006, Art. no. 012340.

[83] J. Haah, "Local stabilizer codes in three dimensions without string logical operators," *Phys. Rev. A, Gen. Phys.*, vol. 83, no. 4, Apr. 2011, Art. no. 042330.

[84] C. Chamberland, G. Zhu, T. J. Yoder, J. B. Hertzberg, and A. W. Cross, "Topological and subsystem codes on low-degree graphs with flag qubits," *Phys. Rev. X*, vol. 10, no. 1, Jan. 2020, Art. no. 011022.

[85] N. P. Breuckmann and J. N. Eberhardt, "Quantum low-density parity-check codes," *PRX Quantum*, vol. 2, no. 4, Oct. 2021, Art. no. 040101.

[86] Q. Xu, J. P. Bonilla Ataides, C. A. Pattison, N. Raveendran, D. Bluvstein, J. Wurtz, B. Vasić, M. D. Lukin, L. Jiang, and H. Zhou, "Constant-overhead fault-tolerant quantum computation with reconfigurable atom arrays," *Nature Phys.*, vol. 20, no. 7, pp. 1084–1090, Jul. 2024.

[87] S. Bravyi, A. W. Cross, J. M. Gambetta, D. Maslov, P. Rall, and T. J. Yoder, "High-threshold and low-overhead fault-tolerant quantum memory," *Nature*, vol. 627, no. 8005, pp. 778–782, Mar. 2024.

[88] P. Panteleev and G. Kalachev, "Degenerate quantum LDPC codes with good finite length performance," *Quantum*, vol. 5, p. 585, Nov. 2021.

[89] A. Leverrier and G. Zémor, "Quantum tanner codes," in *Proc. IEEE 63rd Annu. Symp. Found. Comput. Sci. (FOCS)*, Oct. 2022, pp. 872–883.

[90] V. Paul Su, C. Cao, H.-Y. Hu, Y. Yanay, C. Tahan, and B. Swingle, "Discovery of optimal quantum error correcting codes via reinforcement learning," 2023, *arXiv:2305.06378*.

[91] H. Goto, "Many-hypercube codes: High-rate quantum error-correcting codes for high-performance fault-tolerant quantum computing," 2024, *arXiv:2403.16054*.

[92] L. Chen, H.-X. Li, Y. Lu, C. W. Warren, C. J. Križan, S. Kosen, M. Rommel, S. Ahmed, A. Osman, J. Biznárová, A. F. Roudsari, B. Lienhard, M. Caputo, K. Grigoras, L. Grönberg, J. Govenius, A. F. Kockum, P. Delsing, J. Bylander, and G. Tancredi, "Transmon qubit readout fidelity at the threshold for quantum error correction without a quantum-limited amplifier," *npj Quantum Inf.*, vol. 9, no. 1, p. 26, Mar. 2023.

[93] T. C. van Thiel, M. J. Weaver, F. Berto, P. Duivestein, M. Lemang, K. L. Schuurman, M. Žemlička, F. Hijazi, A. C. Bernasconi, C. Ferrer, E. Cataldo, E. Lachman, M. Field, Y. Mohan, F. K. de Vries, C. C. Bultink, J. C. van Oven, J. Y. Mutus, R. Stockill, and S. Gröblacher, "Optical readout of a superconducting qubit using a piezo-optomechanical transducer," *Nature Phys.*, vol. 21, no. 3, pp. 401–405, Mar. 2025.

[94] J. Tuorila, M. Partanen, T. Ala-Nissila, and M. Möttönen, "Efficient protocol for qubit initialization with a tunable environment," *npj Quantum Inf.*, vol. 3, no. 1, p. 27, Jul. 2017.

[95] R. J. Spiteri, M. Schmidt, J. Ghosh, E. Zahedinejad, and B. C. Sanders, "Quantum control for high-fidelity multi-qubit gates," *New J. Phys.*, vol. 20, no. 11, Nov. 2018, Art. no. 113009.

[96] A. G. Radnaev et al., "A universal neutral-atom quantum computer with individual optical addressing and non-destructive readout," 2024, *arXiv:2408.08288*.

[97] S. J. Evered, D. Bluvstein, M. Kalinowski, S. Ebadi, T. Manovitz, H. Zhou, S. H. Li, A. A. Geim, T. T. Wang, N. Maskara, H. Levine, G. Semeghini, M. Greiner, V. Vuletić, and M. D. Lukin, "High-fidelity parallel entangling gates on a neutral-atom quantum computer," *Nature*, vol. 622, no. 7982, pp. 268–272, Oct. 2023.

[98] S. Zhang, J. Shi, Z. Cui, Y. Wang, Y. Wu, L. Duan, and Y. Pu, "Realization of a programmable multipurpose photonic quantum memory with over-thousand qubit manipulations," *Phys. Rev. X*, vol. 14, no. 2, Apr. 2024, Art. no. 021018.

[99] N. Gisin, "How far can one send a photon?" *Frontiers Phys.*, vol. 10, no. 6, pp. 1–8, Dec. 2015.

[100] T. P. Harty, D. T. C. Allcock, C. J. Ballance, L. Guidoni, H. A. Janacek, N. M. Linke, D. N. Stacey, and D. M. Lucas, "High-fidelity preparation, gates, memory, and readout of a trapped-ion quantum bit," *Phys. Rev. Lett.*, vol. 113, no. 22, Nov. 2014, Art. no. 220501.

[101] T. Ruster, C. T. Schmiegelow, H. Kaufmann, C. Warschburger, F. Schmidt-Kaler, and U. G. Poschinger, "A long-lived zeeman trapped-ion qubit," *Appl. Phys. B*, vol. 122, no. 10, p. 254, Oct. 2016.

[102] X. Shi, J. Sinanan-Singh, K. DeBry, S. L. Todaro, I. L. Chuang, and J. Chiaverini, "Long-lived metastable-qubit memory," *Phys. Rev. A, Gen. Phys.*, vol. 111, no. 2, Feb. 2025, Art. no. 020601.

[103] J. J. Pla, K. Y. Tan, J. P. Dehollain, W. H. Lim, J. J. L. Morton, F. A. Zwanenburg, D. N. Jamieson, A. S. Dzurak, and A. Morello, "High-fidelity readout and control of a nuclear spin qubit in silicon," *Nature*, vol. 496, no. 7445, pp. 334–338, Apr. 2013.

[104] J. Y. Huang et al., "High-fidelity spin qubit operation and algorithmic initialization above 1 K," *Nature*, vol. 627, no. 8005, pp. 772–777, Mar. 2024.

[105] M. Rimbach-Russ, S. G. J. Philips, X. Xue, and L. M. K. Vandersypen, "Simple framework for systematic high-fidelity gate operations," *Quantum Sci. Technol.*, vol. 8, no. 4, Oct. 2023, Art. no. 045025.

[106] S. J. K. Lang, T. Mayer, J. Weber, C. Dhieb, I. Eisele, W. Lerch, Z. Luo, C. Moran Guizan, E. Music, L. Sturm-Rogon, D. Zahn, R. N. Pereira, and C. Kutter, "Advancing superconducting qubits: CMOS-compatible processing and room temperature characterization for scalable quantum computing beyond 2D architectures," 2025, *arXiv:2504.18173*.

[107] Y. Chew, T. Tomita, T. P. Mahesh, S. Sugawa, S. de Léséleuc, and K. Ohmori, "Ultrafast energy exchange between two single Rydberg atoms on a nanosecond timescale," *Nature Photon.*, vol. 16, no. 10, pp. 724–729, Oct. 2022.

[108] C. L. Liu, D.-J. Zhang, X.-D. Yu, Q.-M. Ding, and L. Liu, "A new coherence measure based on fidelity," *Quantum Inf. Process.*, vol. 16, no. 8, pp. 1–10, Aug. 2017.

[109] D. Koch, A. Torrance, D. Kinghorn, S. Patel, L. Wessing, and P. M. Alsing, "Simulating quantum algorithms using fidelity and coherence time as principle models for error," 2019, *arXiv:1908.04229*.

[110] P. Beame and N. Kornerup, "Cumulative memory lower bounds for randomized and quantum computation," 2023, *arXiv:2301.05680*.

[111] B. Schroeder, E. Pinheiro, and W.-D. Weber, "Dram errors in the wild: A large-scale field study," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 37, no. 1, pp. 193–204, 2009.

[112] B. Foxen et al., "Demonstrating a continuous set of two-qubit gates for near-term quantum algorithms," *Phys. Rev. Lett.*, vol. 125, no. 12, Sep. 2020, Art. no. 120504.

[113] M. C. Smith, A. D. Leu, K. Miyanishi, M. F. Gely, and D. Lucas, "Single-qubit gates with errors at the 10^{-7} level," *Phys. Rev. Lett.*, vol. 134, no. 23, 2024, Art. no. 230601.

[114] R. Acharya et al., "Suppressing quantum errors by scaling a surface code logical qubit," *Nature*, vol. 614, no. 7949, pp. 676–681, 2023.

[115] W. K. Wootters and W. H. Zurek, "A single quantum cannot be cloned," *Nature*, vol. 299, no. 5886, pp. 802–803, Oct. 1982.

[116] S. Maurya, A. Molavi, A. Albarghouthi, and S. Tannu, "Managing classical processing requirements for quantum error correction," 2024, *arXiv:2406.17995*.

[117] T. J. Dell, "A white paper on the benefits of chipkill-correct ecc for pc server main memory," *IBM Microelectron. Division*, vol. 11, nos. 1–23, pp. 5–7, 1997.

[118] A. Miranskyy, M. Khan, and U. Mendes, "Comparing algorithms for loading classical datasets into quantum memory," 2024, *arXiv:2407.15745*.

[119] I. Pietikäinen, O. Černotík, A. Eickbusch, A. Maiti, J. W. O. Garmon, R. Filip, and S. M. Girvin, "Strategies and trade-offs for controllability and memory time of ultra-high-quality microwave cavities in circuit QED," 2024, *arXiv:2403.02278*.

[120] M.-H. Nguyen et al., "Cryogenic memory architecture integrating spin Hall effect based magnetic memory and superconductive cryotron devices," *Sci. Rep.*, vol. 10, no. 1, p. 248, Jan. 2020.

[121] D. S. Holmes, "Cryogenic electronics and quantum information processing," in *Proc. IEEE Int. Roadmap Devices Syst. Outbriefs*, Nov. 2021, pp. 1–93.

[122] C. J. Ballance, T. P. Harty, N. M. Linke, M. A. Sepiol, and D. M. Lucas, "High-fidelity quantum logic gates using trapped-ion hyperfine qubits," *Phys. Rev. Lett.*, vol. 117, no. 6, Aug. 2016, Art. no. 060504.

[123] K. T. Malladi, B. C. Lee, F. A. Nothaft, C. Kozyrakis, K. Periyathambi, and M. Horowitz, "Towards energy-proportional datacenter memory with mobile DRAM," *ACM SIGARCH Comput. Archit. News*, vol. 40, no. 3, pp. 37–48, Sep. 2012.

[124] K. Zaman, A. Marchisio, M. Abdullah Hanif, and M. Shafique, "A survey on quantum machine learning: Current trends, challenges, opportunities, and the road ahead," 2023, *arXiv:2310.10315*.

[125] S. S. Cranganore, V. De Maio, I. Brandic, and E. Deelman, "Paving the way to hybrid quantum–classical scientific workflows," *Future Gener. Comput. Syst.*, vol. 158, pp. 346–366, Sep. 2024.

[126] F. Phillipson, N. M. P. Neumann, and R. Wezeman, "Classification of hybrid quantum-classical computing," in *Proc. Int. Conf. Comput. Sci.*, 2023, pp. 18–33.

[127] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles, "Variational quantum algorithms," *Nature Rev. Phys.*, vol. 3, no. 9, pp. 625–644, 2021.

[128] M. Suchara, Y. Alexeev, F. T. Chong, H. Finkel, H. Hoffmann, J. Larson, J. C. Osborn, and G. A. Smith, "Hybrid quantum-classical computing architectures," in *Proc. 3rd Int. Workshop Post-Moore Era Supercomputing*, 2018, pp. 1–3.

[129] B. Weder, J. Barzen, M. Beisel, and F. Leymann, "Analysis and rewrite of quantum workflows: Improving the execution of hybrid quantum algorithms," in *Proc. Closer*, 2022, pp. 38–50.

[130] D. Monroe, "Building a practical quantum computer," *Commun. ACM*, vol. 65, no. 7, pp. 15–17, Jul. 2022.

[131] P. Lamichhane and D. B. Rawat, "Quantum machine learning: Recent advances, challenges, and perspectives," *IEEE Access*, vol. 13, pp. 94057–94105, 2025.
[132] M. Schlosshauer, "Quantum decoherence," *Phys. Rep.*, vol. 831, pp. 1–57, Oct. 2019.
[133] W. G. Ritter, "Quantum channels and representation theory," *J. Math. Phys.*, vol. 46, no. 8, pp. 082103-1–082103-22, Aug. 2005.
[134] D. Jiao and Y. Li, "Fault-tolerant fidelity based on few-qubit codes: Parity-check circuits for biased error channels," 2020, *arXiv:2009.09726*.
[135] D. Aharonov and M. Ben-Or, "Fault-tolerant quantum computation with constant error," in *Proc. Annu. ACM Symp. Theory Comput.*, 1997, pp. 176–188.
[136] H. Kurniawan, L. Rodríguez-Soriano, D. Cuomo, C. G. Almudever, and F. G. Herrero, "On the use of calibration data in error-aware compilation techniques for NISQ devices," in *Proc. IEEE Int. Conf. Quantum Comput. Eng. (QCE)*, Sep. 2024, pp. 338–348.
[137] T. Giurgica-Tiron, Y. Hindy, R. LaRose, A. Mari, and W. J. Zeng, "Digital zero noise extrapolation for quantum error mitigation," *IEEE Int. Conf. Quantum Comput. Eng. (QCE)*, Sep. 2020, pp. 306–316.
[138] P. Das, S. Tannu, S. Dangwal, and M. Qureshi, "ADAPT: Mitigating idling errors in qubits via adaptive dynamical decoupling," in *Proc. 54th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Oct. 2021, pp. 950–962.
[139] A. Khan and F. Ahmed, "Future prospects of quantum machine learning in accelerating drug discovery processes," *Emerg. Trends Mach. Intell. Big Data*, vol. 15, no. 10, pp. 50–66, 2023.
[140] N. P. Landsman, "Born rule and its interpretation," in *Compendium of Quantum Physics*, 2009, pp. 64–70.
[141] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, "The theory of variational hybrid quantum-classical algorithms," *New J. Phys.*, vol. 18, no. 2, Feb. 2016, Art. no. 023023.
[142] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," 2014, *arXiv:1411.4028*.
[143] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, "Supervised learning with quantum-enhanced feature spaces," *Nature*, vol. 567, no. 7747, pp. 209–212, Mar. 2019.
[144] S. L. Tsang, M. T. West, S. M. Erfani, and M. Usman, "Hybrid quantum–classical generative adversarial network for high-resolution image generation," *IEEE Trans. Quantum Eng.*, vol. 4, pp. 1–19, 2023.

PRADEEP LAMICHHANE received the B.S. degree in computer science from Howard University, Washington, DC, USA, where he was recognized as a Google Tech Exchange Scholar. He is currently pursuing the Ph.D. degree in computer science with Howard University. During his academic journey, he completed internships as a Sophomore Summer Analyst with the Bank of America and a Software Engineering Intern with Microsoft. He was a tutor with the Mathematics Department, Howard University, for three years. He was also a Research Assistant with Capital One and Howard University, contributing to various projects in his field of interest. He is a Research Assistant with Howard University. His research interests include quantum computing and quantum machine learning.

DANDA B. RAWAT (Senior Member, IEEE) received the Ph.D. degree from Old Dominion University, Norfolk, VA, USA. He was the Graduate Program Director of Howard Computer Science Graduate Programs, from 2017 to 2025. He is currently the Associate Dean for Research and Graduate Studies, a Full Professor with the Department of Electrical Engineering and Computer Science (EECS), the Founding Director of the Data Science and Cybersecurity Center, Howard University, Washington, DC, USA, and the DoD Center of Excellence in Artificial Intelligence and Machine Learning (CoE-AIML), and the Director of the Cyber-Security and Wireless Networking Innovations (CWiNs) Research Laboratory and the Graduate Cybersecurity Certificate Program, Howard University. He successfully led and established the Research Institute for Tactical Autonomy (RITA), the 15th University Affiliated Research Center (UARC) of U.S. Department of Defense, as the PI/Founding Executive Director of Howard University, Washington, DC, USA. He has delivered over 100 keynotes and invited speeches at international conferences and workshops. He has published over 350 scientific/technical articles and 11 books. He has successfully supervised and graduated 40 Ph.D. students (out of which 31 were underrepresented Ph.D. students, including 14 female Ph.D. students), successfully supervised more than 30 M.S. students and mentored eight postdocs, and has been supervising 22 Ph.D. students and mentoring three postdocs. Furthermore, he has successfully mentored over 120 minority undergraduate students. He is engaged in research and teaching in the areas of cybersecurity, machine learning, big data analytics, and wireless networking for emerging networked systems, including cyber-physical systems, the Internet-of-Things, multi domain operations, smart cities, software defined systems, and vehicular networks. He has secured over $110 million as a PI and over $18 million as a Co-PI in research funding from U.S. National Science Foundation (NSF), U.S. Department of Homeland Security (DHS), U.S. National Security Agency (NSA), U.S. Department of Energy, National Nuclear Security Administration (NNSA), National Institute of Health (NIH), U.S. Department of Defense (DoD) and DoD Research Labs, Industry (Microsoft, Intel, VMware, PayPal, Mastercard, Meta, BAE, and Raytheon), and private foundations. He served as a Technical Program Committee (TPC) Member for several international conferences, including IEEE INFOCOM, IEEE GLOBECOM, IEEE CCNC, IEEE GreenCom, IEEE ICC, IEEE WCNC, and IEEE VTC conferences. He is a Lifetime Professional Senior Member of ACM, a Lifetime Member of the Association for the Advancement of Artificial Intelligence (AAAI), a Champion Member of USENIX (Advanced Computing Systems Association), a Lifetime Member of SPIE, a Member of ASEE, AAAS, and the SAE International, and a fellow of the Institution of Engineering and Technology (IET). He was a recipient of U.S. NSF CAREER Award, U.S. Department of Homeland Security (DHS) Scientific Leadership Award, the Presidents' Medal of Achievement Award (2023) at Howard University, the Provost's Distinguished Service Award 2021, the Researcher Exemplar Award 2019 and the Graduate Faculty Exemplar Award 2019 from Howard University, U.S. Air Force Research Laboratory (AFRL) Summer Faculty Visiting Fellowship 2017, the Outstanding Research Faculty Award (Award for Excellence in Scholarly Activity) at GSU in 2015, and the Best Paper Award (IEEE CCNC, IEEE ICII, IEEE DroneCom, and BWCA). He served as the Vice Chair for the Executive Committee of the IEEE Savannah Section from 2013 to 2017. He has been serving as an Editor/Guest Editor for over 100 international journals, including an Associate Editor for IEEE TRANSACTIONS ON BIG DATA, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING, IEEE TRANSACTIONS ON SERVICES COMPUTING (2018–2022), IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING (2019–2023); and *ACM Transactions on Intelligent Systems and Technology*; an Editor for IEEE INTERNET OF THINGS JOURNAL and IEEE COMMUNICATIONS LETTERS; and a Technical Editor for IEEE NETWORK. He has been in the Organizing Committees for several IEEE flagship conferences, such as IEEE INFOCOM, IEEE CNS, IEEE ICC, and IEEE GLOBECOM. He is an ACM Distinguished Speaker and an IEEE Distinguished Lecturer (FNTC and VTS). More info can be found at https://www.rawat.info/bio.