Origin Compass: A Quantum Operating System for Efficient Quantum Resource Utilization

Authors: Weicheng Kong, Wang Junchao, Han Yongjian, Wu Yuchun, Zhang Yu, Dou Menghan, square and circle, Guo Guoping, Guoping Guo

Date: 2021-05-17T00:00:00+00:00

Abstract

With the continuous advancement of quantum processor technologies such as superconducting quantum processors, the efficient utilization of quantum processors and the effective collaboration between quantum computers and classical computing platforms have become urgent requirements for the practical development of quantum computing. Benyuan Sinan is a quantum operating system designed to meet this demand, providing services such as quantum task scheduling, quantum resource management, quantum program compilation, and automated qubit calibration, thereby uniformly and efficiently managing quantum computing resources. This paper proposes technical solutions for Benyuan Sinan, including multi-quantum-processor load balancing, multi-quantum-program parallel computing based on quantum circuit mapping, and automated qubit calibration based on hidden Markov chains. Through a comparison of data before and after connecting quantum computing platforms such as Benyuan Wuyuan to Benyuan Sinan: in experiments testing four representative quantum circuits (QFT, GHZ, DJ, BV), the fidelity of circuits mapped by Benyuan Sinan improved by at least 10% compared to the mapping results of the BMT algorithm; in experiments running the GHZ quantum circuit, the single-quantum-processor parallel computing and multi-quantum-processor load balancing capabilities provided by Benyuan Sinan improved the operational efficiency of quantum processors by at least 120%. Benyuan Sinan can collaboratively schedule computing resources such as quantum processors (QPU) and classical computers, providing effective resource management for the widespread use of quantum computing.

Full Text

Preamble

Origin Pilot: A Quantum Operating System for Efficient Utilization of Quantum Resources

Wei-Cheng Kong[1,2], Jun-Chao Wang[2,4], Yong-Jian Han[2], Yu-Chun Wu[2], Yu Zhang[3], Meng-Han Dou[1], Yuan Fang[1], Guo-Ping Guo[1,2]

[1]Origin Quantum Computing Company Limited, Hefei 230026, China
[2]CAS Key Laboratory of Quantum Information (University of Science and Technology of China), Hefei 230026, China
[3]School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China
[4]State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450002, China

Corresponding Author: Guo-Ping Guo, E-mail: gpguo@ustc.edu.cn

Abstract

With continuous advances in quantum processor technologies such as superconducting systems, the efficient utilization of quantum processors and effective collaboration between quantum and classical computing platforms have become urgent requirements for the practical development of quantum computing. Origin Pilot is a quantum operating system designed to address these needs. It provides services including quantum task scheduling, quantum resource management, quantum program compilation, and automated qubit calibration, enabling unified and efficient management of quantum computing resources. This paper proposes technical solutions for Origin Pilot, including multi-quantum processor load balancing, multi-quantum program parallel computing based on quantum circuit mapping, and automated qubit calibration based on implicit Markov chains. By comparing data before and after integrating the Wuyuan quantum computing platform with Origin Pilot, we demonstrate that for four representative quantum circuits (QFT, GHZ, DJ, BV), the fidelity of circuits mapped by Origin Pilot improves by at least 10% compared to the BMT algorithm. In experiments running GHZ circuits, Origin Pilot' s single-processor parallel computing and multi-processor load balancing capabilities increase quantum processor operational efficiency by at least 120%. Origin Pilot can coordinate quantum processors (QPUs), classical computers, and other computing resources, providing effective resource management for widespread quantum computing adoption.

Keywords: Origin Pilot; Quantum Operating System; Quantum Computing; Quantum Processor

1 Introduction

Quantum computing represents a novel computational paradigm that combines quantum mechanics with computer science. It uses quantum bits as basic computational units and leverages quantum properties such as superposition and entanglement to achieve exponential speedup over classical computing for specific problems. Quantum computing demonstrates computational advantages over classical methods in numerous fields, including asymmetric cryptography, quantum chemistry, finance, and machine learning.

Currently, the primary physical implementations of quantum computing include superconducting systems, semiconductor spin systems, ion trap systems, and optical systems. With improvements in both hardware (such as material and process enhancements, environmental noise optimization, and control electronics development) and software (including quantum control architectures and basic quantum software), preliminary applications of quantum computing have become feasible. On one hand, Google demonstrated "quantum supremacy" in 2019 by validating superior computational capability on a 53-qubit Sycamore chip for random circuit sampling problems. On the other hand, cloud-based quantum computing services from providers including IBM, D-Wave, Google, Rigetti, and Origin Quantum have begun serving diverse tasks and accumulating non-specialist users and researchers.

However, as quantum computing resources, demands, and general users grow rapidly, managing quantum computing devices and efficiently utilizing quantum resources have become key obstacles. Henry Corrigan-Gibbs et al. first proposed the concept of a Quantum Operating System, describing three potential underlying hardware architectures: quantum FPGA (attached to classical CPU), quantum x86 (implementing classical x86 instruction sets plus universal quantum instruction sets), and quantum distributed systems. Reid Honan et al. constructed a serializable conflict graph by analyzing relationships between quantum programs, qubit mapping, and connectivity, proposing a method for parallel execution of multiple quantum programs on a single quantum processor. To shield application developers from physical details of underlying hardware, UK' s Riverlane released Deltaflow.OS, while Austria' s ParityQC released ParityOS for compilation optimization on specific quantum processors. Despite these solutions, challenges remain in multi-quantum processor scheduling and automated qubit calibration and optimization.

Multi-quantum processor scheduling: Current cloud-based quantum computers require users to manually select quantum processors rather than having the system automatically allocate tasks based on availability and feasibility. This leads to localized resource congestion while other resources remain idle. Efficient automatic resource allocation and scheduling across multiple processors based on resource consumption is needed.

Automated qubit calibration and optimization: Qubits are vulnerable to environmental noise and performance instability, reducing quantum gate fidelity and causing incorrect computation results. Current offline periodic calibration suffers from two problems: (a) qubit quality cannot be guaranteed between calibration intervals, and (b) well-functioning qubits cannot provide computational services during calibration, wasting resources. Google developed an on-demand automated calibration system called Optimus, though its online calibration capabilities remain unclear.

To address these challenges, this paper proposes Origin Pilot, a quantum operating system that efficiently utilizes quantum resources. Through four core services—quantum thread scheduling, automated qubit calibration and optimiza-

tion, quantum program compilation, and quantum resource management—Origin Pilot effectively solves these problems and improves quantum resource utilization efficiency.

The remainder of this paper is organized as follows: Section 2 introduces basic concepts of quantum computing and quantum operating systems. Section 3 presents the overall architecture and workflow of Origin Pilot. Section 4 details solutions for multi-quantum processor load balancing, multi-quantum program parallel computing, and automated qubit calibration. Section 5 provides experimental validation. Section 6 discusses future trends and work.

2.1 Quantum Computing

A quantum bit is the fundamental unit of quantum computing. Unlike classical bits that can only represent 0 or 1 at any time, a quantum bit can exist in a superposition state: $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$. Quantum measurement collapses the quantum state to a classical outcome with specific probabilities.

Quantum logic gates are unitary transformations acting on quantum bits that implement state transformations. Unlike classical logic gates, quantum gates are reversible. Universal quantum computing requires only single-qubit unitary transformations and two-qubit CNOT gates. Specifically, single-qubit gates can be generated by Hadamard, T, and S gates. Common quantum logic gates include:

Single-qubit gates:

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}, \quad S = \begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix}, \quad T = \begin{pmatrix}1 & 0\\ 0 & e^{i\pi/4}\end{pmatrix}$$

Multi-qubit gates:

$$CNOT = \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix}$$

The quantum circuit model is the most commonly used quantum computing model. In this model, any unitary transformation can be implemented through ordered combinations of universal quantum gates, which we call a quantum circuit (similar to classical gate circuits, but with quantum gates). Quantum circuits can be visualized as circuit diagrams, as shown in Figure 1 [Figure 1: see original paper].

Generally, quantum circuits input the quantum state $|0\rangle^{\otimes n}$, and measurement at the end yields computational results. A quantum program is an ordered sequence of quantum gates, quantum measurements, and classical computations that may include multiple quantum circuit segments.

Quantum State Fidelity Due to environmental noise and quantum processor imperfections, actual execution results often deviate from ideal outcomes. This deviation can be measured using fidelity between ideal and actual quantum states:

$$F(\rho, \sigma) = \left(\mathrm{Tr} \sqrt{\sqrt{\rho} \sigma \sqrt{\rho}} \right)^2$$

Higher fidelity indicates smaller deviation and better computational results.

2.2 Differences Between Classical and Quantum Computers

Quantum and classical computing follow different physical laws (quantum mechanics vs. classical mechanics), leading to fundamental differences in architecture and computation methods. Key distinctions include:

Qubit Mapping: Quantum processors have different topologies and support different native gates depending on fabrication processes. This makes qubit mapping efficiency vary dramatically between schemes. Additionally, each qubit' s real-time state differs, adding complexity. This represents a fundamental difference from classical bit addressing.

Quantum Parallel Computing: Threads are the basic scheduling units in classical operating systems, with single-core CPUs executing only one thread at a time. Multiple threads achieve concurrency through context switching. In quantum computing, quantum programs are the minimal execution units. Due to quantum characteristics, quantum programs cannot be interrupted and switched like classical programs. However, multiple quantum programs using different physical qubits can execute in parallel on the same quantum processor.

Automated Qubit Calibration: Classical chips have mature fabrication and stable bit performance without short-term fluctuations, so classical OS focuses on resource utilization optimization like memory management. Quantum processors suffer from performance degradation over time due to environmental effects. Fixed error rates for single-qubit, two-qubit, and readout operations yield invalid results. Only through continuous qubit property monitoring and automated calibration can quantum OS ensure correctness. Stable, high-quality qubit performance is prerequisite for multi-program parallelism and efficient resource utilization.

In summary, quantum computers require specialized qubit mapping, quantum parallel computing, and qubit calibration—needs unmet by current classical operating systems, making quantum OS development imperative.

2.3 Basic Definitions in Origin Pilot Quantum OS

Quantum Application: A quantum-classical hybrid program designed to complete one or more computational tasks.

Quantum Task: The basic task unit sent by a quantum application, represented as a quantum program. A quantum application may contain multiple quantum programs, with results processed by classical algorithms.

Quantum Transaction: The basic execution unit on a single quantum processor. It can contain multiple quantum tasks that execute atomically—all tasks complete or none do.

Quantum Thread: The basic scheduling unit in the quantum OS. The OS schedules based on resources required by quantum transactions. A quantum transaction becomes a quantum thread when assigned to a quantum processor, serving as its execution vehicle.

Quantum Processor: The execution unit for quantum threads. A quantum processor can execute only one quantum thread at a time, though multiple processors may be used within one quantum application.

Quantum Resource: Physical systems following quantum mechanics laws for quantum information processing and storage, with the quantum bit as the basic unit. Resources include quantum processors, quantum memory, etc.

3.1 Overall Architecture

Origin Pilot adopts a quantum-classical hybrid architecture supporting multiple backends including quantum processors, quantum virtual machines, and high-performance computing clusters. Quantum computers require classical assistance (e.g., verifying quantum results for NP problems), and hybrid algorithms like quantum machine learning, quantum chemistry, and quantum finance play important roles. Therefore, complete quantum applications require both quantum task execution and classical information processing.

Origin Pilot classifies services as quantum or classical based on whether they involve quantum computation, with coordinated interaction between them (see Figure 2 [Figure 2: see original paper]).

Quantum Services handle quantum tasks and interface with quantum backends, including quantum task scheduling, quantum resource management, automated qubit calibration, and quantum program compilation. These services improve fidelity of parallel quantum computation and qubit utilization.

Classical Services handle classical computation, interfacing with classical backends while managing quantum environment loading, system monitoring, and device status. Services include job scheduling, system monitoring, logging, quantum computer configuration, and device monitoring. These address large-scale classical data processing needs in hybrid algorithms and ensure system stability.

Quantum applications invoke these services through quantum programming frameworks and distributed computing frameworks to achieve heterogeneous quantum-classical distributed computing. Applications generate input data via

classical services, create quantum programs, submit quantum tasks to quantum services, and analyze results through classical services to produce subsequent inputs. Users can also manage quantum devices and resources through the OS resource manager.

3.2 Origin Pilot Workflow

The Origin Pilot workflow is illustrated in Figure 3 [Figure 3: see original paper]:

1. Users write quantum-classical hybrid applications in QRunes language. The quantum compiler performs lexical, syntactic, and semantic analysis to identify classical and quantum components, translating and optimizing the quantum portion into quantum gates. It then compiles the application into a quantum-classical hybrid executable submitted to the Origin Pilot server. For applications requiring high-performance classical clusters, users can employ distributed computing frameworks for classical code.
2. Upon receiving the executable, Origin Pilot runs it. Classical computation executes on the host server or, if using distributed frameworks, is dispatched to high-performance clusters via the job scheduling system.
3. The quantum program portion submits its OriginIR (quantum intermediate representation) to the quantum task scheduling service. The scheduler sorts tasks by priority and combines multiple high-priority tasks meeting resource requirements into a quantum transaction. This transaction undergoes qubit mapping to adapt to processor topology, then low-level compilation to target instruction sets, producing executable programs and pulse waveforms. The transaction is then sent to quantum devices. Before execution, it binds to a quantum thread specifying the target processor, recording transaction ID, processor ID, task IDs, executable program, and occupied qubits. Results return via the quantum thread to the corresponding tasks and hybrid application.

During quantum task execution, Origin Pilot continuously monitors and calibrates quantum resources through the automated calibration service. The calibration program periodically detects qubit states. If states fall below computational requirements, the service notifies resource management to mark qubits as pending calibration and sends calibration tasks to the scheduler at highest priority. Calibration tasks are combined with quantum tasks into transactions for execution.

4 Solutions Provided by Origin Pilot

Having described the overall architecture and workflow, this section details solutions for multi-quantum processor load balancing, multi-quantum program parallel computing, and automated qubit calibration.

4.1 Multi-Quantum Processor Load Balancing

Multi-quantum processor load balancing is the core problem for quantum task scheduling. To understand the scheduling process, we first introduce quantum task components: (1) required qubit count, (2) quantum program intermediate representation, (3) quantum processor ID, (4) task type (quantum program or calibration), and (5) priority.

If element (3) specifies processor A, the task can only run on A. If unspecified, the scheduler allocates based on system state. Different scheduling algorithms apply to quantum programs versus calibration tasks.

Calibration tasks require fast response (real-time), short execution, and explicit qubit targeting. They receive highest priority to ensure computational reliability. Quantum programs need not be real-time and can be efficiently allocated across multiple processors based on available resources. For these tasks, Origin Pilot employs a quantum high-response-ratio-priority scheduling algorithm that considers both estimated service time (predicted by the quantum compiler) and waiting time. The response ratio is defined as:

$$R_p = \frac{\text{waiting time} + \text{required service time}}{\text{required service time}}$$

Tasks with larger R_p execute first. This algorithm combines first-come-first-served and shortest-job-first advantages with dynamic priority that increases with waiting time, ensuring long tasks eventually execute.

The scheduling process is shown in Figure 4 [Figure 4: see original paper]. Upon receiving a quantum task, the scheduler: (1) obtains its required service time, calculates $R_p = 1$, and adds it to the waiting queue; (2) continuously updates R_p as waiting time increases and reorders the queue; (3) selects multiple high-priority tasks whose total qubit requirement does not exceed the largest available region across processors, merging them into a quantum transaction for balanced allocation; (4) compiles the transaction into executable programs and pulses; (5) binds to a quantum thread for execution.

Notably, calibration tasks always appear at the queue front and join transactions without participating in qubit mapping (their target qubits are predetermined). Two key points: (1) processors are divided into execution and calibration regions (see Section 4.3), so multiple calibration tasks only affect calibration regions without starving quantum programs; (2) the algorithm follows first-come-first-served among calibration tasks, with priority determined by arrival order.

4.2 Multi-Quantum Program Parallel Computing

Origin Pilot currently implements synchronous parallel quantum computing: multiple quantum tasks form a quantum transaction that starts simultaneously,

with the next transaction executing only after all tasks complete. Before execution, logical qubits must be mapped to physical qubits to minimize SWAP gates needed for non-nearest-neighbor operations.

The synchronous parallel workflow is shown in Figure 4. The qubit mapping algorithm simultaneously optimizes SWAP gate count and noise-aware fidelity for non-nearest-neighbor gates. The implementation: (1) converts the quantum program to a directed acyclic graph (DAG) where vertices represent two-qubit gates, edges indicate shared qubits, and direction shows temporal order; (2) extracts vertices with indegree 0 and traverses, selecting directly mappable vertices (requiring no SWAPs) and recording each mapping as a subgraph; (3) removes mapped vertices and edges, obtains a new DAG, and extends subgraphs from indegree-0 vertices, deleting non-extendable subgraphs iteratively until no indegree-0 vertices are directly mappable, yielding a maximal subgraph set; (4) repeats to obtain multiple maximal subgraph sequences; (5) uses Token-Swapping to compute minimal-cost paths between adjacent subgraphs, connecting them to produce multiple mapping schemes; (6) calculates overall fidelity for each scheme and selects the optimal one.

This algorithm improves real-chip execution correctness by considering path fidelity information. The pseudocode is:

```
Input: src_{QProg} (original quantum circuit), QPU_{adj} (physical quantum chip topology)
Output: mapped_{QC} (mapped circuit conforming to physical topology)

Convert src_{QProg} to DAG
Initialize 2D structure sub_{graph}_{vec}_{2d} for maximal subgraph sequences

// Phase 1: Traverse DAG to obtain maximal subgraph sequences
while DAG vertex count > 0:
    Extract indegree-0 vertices V
    if subgraph sequence S is not empty:
        if all subgraphs in S cannot be extended:
            Store S as maximal subgraph sequence in sub_{graph}_{vec}_{2d}
            Clear S
            break
    Extend S based on QPU_{adj}, deleting non-extendable subgraphs
    Initialize S based on V mapping possibilities on QPU_{adj}

// Phase 2: Token-Swapping to find optimal path
Initialize best_{path}_{vec} for optimal paths
foreach S_{cur} in sub_{graph}_{vec}_{2d}:
    Compute minimal cost from each subgraph in S_{cur} to each in S_{next} via Token-Swappi
    Append optimal paths to sub_{best}_{path}_{vec}
foreach best_{path} in best_{path}_{vec}:
    Calculate overall fidelity
Select highest-fidelity final_{best}_{path}
```

```
// Phase 3: Generate mapped_{QC} from final_{best}_{path}
foreach path_{node} in final_{best}_{path}:
    if path_{node} is subgraph:
        Convert to sub_{cir} and insert into mapped_{QC}
    if path_{node} is swap information:
        Insert swap-gates into mapped_{QC}
return mapped_{QC}
```

4.3 Automated Qubit Calibration

Key performance metrics for qubit quality include coherence time and gate fidelity. Coherence time describes how long quantum states remain coherent and indicates environmental coupling strength—critical since quantum algorithms may require many gates. Gate errors must remain below 1% for universal quantum computing.

Given the complexity of factors affecting qubit quality, continuous calibration is essential. Calibration involves verification and adjustment phases. Verification uses periodic checks and random walks to detect performance parameters and determine calibration needs. Adjustment applies targeted corrections based on error types and magnitudes.

Since performance parameters depend on multiple correlated physical parameters, identifying causes of degradation is key. This process can be modeled as a Markov process for automation. We constructed a partially observable Markov decision process-based automated calibration system enabling on-demand calibration.

Additionally, we designed a real-time online calibration scheme using partitioned automated calibration. Based on verification results, the quantum processor is dynamically divided into execution and calibration regions. The compilation service compiles both calibration and quantum tasks according to these regions, forming quantum transactions that ensure simultaneous execution. This guarantees quantum tasks run on the processor' s best-performing regions while calibration proceeds in parallel.

5 Experiments and Results

To demonstrate Origin Pilot' s effective quantum resource management, we conducted comparative tests.

5.1 Runtime Comparison for Identical Quantum Tasks

All experiments ran on the real Wuyuan quantum processor—Origin Quantum' s self-developed 6-qubit superconducting quantum processor KuaFu KF C6-130 with internationally advanced fidelity and coherence metrics.

5.1.1 Objective and Setup

This experiment compares efficiency before and after Origin Pilot integration when processing identical tasks on multiple quantum processors (Wuyuan #1 and #2). Running GHZ circuits validates Origin Pilot' s support for single-processor parallel computing and multi-processor distributed computing.

5.1.2 Procedure

Without Origin Pilot: Recorded runtime for 10 executions of a 2-qubit GHZ circuit in single-processor, single-circuit mode. With Origin Pilot: Recorded runtime for 10 executions using single-processor multi-circuit parallel, dual-processor single-circuit, and dual-processor multi-circuit parallel modes. Repeated the entire process 10 times.

5.1.3 Results

Runtime distributions are shown in Figure 5 [Figure 5: see original paper].

5.1.4 Conclusion

Origin Pilot' s support for single-processor parallel computing and multi-processor load balancing significantly improves processor utilization, achieving at least 120% efficiency improvement for 2-qubit GHZ circuits.

5.2 Automated Calibration Effects

All experiments ran on the real Wuyuan quantum processor.

5.2.1 Objective

To validate automated calibration effectiveness, we set fidelity thresholds: qubits falling below thresholds cannot provide service. Three comparison experiments were conducted: single-qubit gate fidelity (calibrated vs. uncalibrated), two-qubit gate fidelity (calibrated vs. uncalibrated), and available task count (calibrated vs. uncalibrated).

5.2.2 Setup and Procedure

1. **Single-qubit gate fidelity:** Calibration threshold set to 0.98
2. **Two-qubit gate fidelity:** Calibration threshold set to 0.95
3. **Available task count:** Compared processor capacity with and without calibration service

5.2.3 Results

Results are shown in Figures 6-8 [Figure 6: see original paper][Figure 7: see original paper][Figure 8: see original paper].

5.2.4 Conclusion

Without automated calibration, single- and two-qubit gate fidelities showed monotonic decline. With calibration, fidelities remained above 0.98 and 0.95 respectively. The calibrated processor sustained service capacity, running far more programs than the uncalibrated processor, which became unusable after some time.

5.3 Mapping Optimization Effectiveness

5.3.1 Objective

To validate Origin Pilot' s effectiveness in mapping quantum circuits to superconducting processor topology while considering fidelity, we compared fidelity before and after optimization for four representative quantum programs.

5.3.2 Setup

Experiments used Origin Quantum' s noisy virtual machine configured with: 1. 2×4 grid topology where adjacent qubits support two-qubit gates 2. QPanda noisy VM with decoherence and depolarization noise, considering only CZ gate fidelity 3. Test circuits: QFT, GHZ, DJ, and BV algorithms 4. Quantum State Tomography (QST) to compute fidelity for each mapped circuit

5.3.3 Results

Circuit mappings are shown in Figures 10-13 [Figure 10: see original paper][Figure 11: see original paper][Figure 12: see original paper][Figure 13: see original paper].

5.3.4 Fidelity Comparison

Circuit	BMT Algorithm	Origin Pilot Mapping
QFT	0.1266$±0.004\|0.2139±0.003\|\|$GHZ$\|0.4305±0.005\|0.6032±0.004\|\|DJ|0.4305±0.004|0.8596±0.0	

5.3.5 Conclusion

Origin Pilot achieves better mapping than BMT for all four representative quantum programs, improving circuit fidelity by at least 10%.

6 Future Trends and Outlook

Origin Pilot aims to efficiently utilize precious quantum resources while fully integrating with classical computing. Its services for quantum task scheduling, resource management, program compilation, and automated calibration achieve efficient quantum resource management.

Future work will support user-transparent quantum distributed computing via circuit partitioning, quantum asynchronous parallel computing, and adaptation to different physical quantum processor architectures. We will further improve automated calibration, compilation, and scheduling efficiency, enabling multi-physics quantum distributed computing, hybrid HPC-plus-QPU systems, and QPU-plus-quantum-virtual-machine architectures.

References

[1] Nielsen M A, Chuang I L. Quantum Computation and Quantum Information[J]. Mathematical Structures in Computer Science, 2002, 17(6):1115-1115.
[2] Grover L.K, A fast quantum mechanical algorithm for database search, Proceedings, 28th Annual ACM Symposium on the Theory of Computing, (May 1996) p. 212.
[3] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for linear systems of equations," Phys. Rev. Lett. 103.15 (2009), p.
[4] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," SIAM review, vol. 41, no. 2, pp. 303–332, 1999.
[5] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," Nature, vol.549, no. 7671, pp. 242–246, 2017.
[6] Ghosh B, Kozarević E. Identifying explosive behavioral trace in the CNX nifty index: A quantum finance approach[J]. Investment Management & Financial Innovations, 2018, 15(1): 208.
[7] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," Nature, vol. 549, no. 7671, pp.195–202, 2017.
[8] Schuld M, Sinayskiy I, Petruccione F. An introduction to quantum machine learning[J]. Contemporary Physics, 2015, 56(2):
[9] Houck, A., Türeci, H. & Koch, J. On-chip quantum simulation with superconducting circuits. Nature Phys 8, 292–299 (2012). https://doi.org/10.1038/nphys2251.
[10] Li, Xiaoqin & Wu, Yanwen & Steel, Duncan & Gammon, D & Stievater, Todd & Katzer, D & Park, D. & Piermarocchi, Carlo & Sham, L. (2003). An All-Optical Quantum Gate in a Semiconductor Quantum Dot. Science (New York, N.Y.). 301. 809-11. 10.1126/science.1083800.
[11] H. Häffner, C.F. Roos, R. Blatt, Quantum computing with trapped ions, Physics Reports, Volume 469, Issue 4, 2008, Pages 155-203, ISSN 0370-1573.
[12] Kielpinski, D., Monroe, C. & Wineland, D. Architecture for a large-scale ion-trap quantum computer. Nature 417, 709–711 (2002).
[13] Lvovsky, A., Sanders, B. & Tittel, W. Optical quantum memory. Nature Photon 3, 706–714 (2009).
[14] O' Brien JL. Optical quantum computing. Science. 2007 Dec 7;318(5856):1567-70.
[15] Guo Y, Zhang YF, Bao XY, Han TZ, Tang Z, Zhang LX, Zhu WG, Wang EG, Niu Q, Qiu ZQ, Jia JF, Zhao ZX, Xue QK. Superconductivity modulated

by quantum size effects. Science. 2004 Dec 10;306(5703):1915-7.
[16] Suter, D. and Gonzalo A. Alvarez. "Colloquium: Protecting quantum information against environmental noise." Reviews of Modern Physics 88 (2016): 041001.
[17] Wu R B, Chu B, Owens D H, et al. Data-driven gradient algorithm for high-precision quantum control[J]. Physical Review A, 2018, 97(4): 042122.
[18] Botero U J, Tehranipoor M M, Forte D. Upgrade/downgrade: Efficient and secure legacy electronic system replacement[J]. IEEE Design & Test, 2018, 36(1): 14-22.
[19] Orgiazzi J L, Deng C, Layden D, et al. Flux qubits in a planar circuit quantum electrodynamics architecture: quantum control and decoherence[J]. Phys.rev.b, 2016, 93(10):104518.
[20] LaRose R. Overview and comparison of gate level quantum software platforms[J]. Quantum, 2019, 3: 130.
[21] Alexeev Y, Bacon D, Brown K R, et al. Quantum computer systems for scientific discovery[J]. PRX Quantum, 2021, 2(1):
[22] Kottmann J S, Alperin-Lea S, Tamayo-Mendoza T, et al. Tequila: A platform for rapid development of quantum algorithms[J]. Quantum Science and Technology, 2021, 6(2): 024009.
[23] Arute, F., Arya, K., Babbush, R. et al. Quantum supremacy using a programmable superconducting processor. Nature 574, 505–510 (2019). https://doi.org/10.1038/s41586-019-1666-5.
[24] Harrow, A., Montanaro, A. Quantum computational supremacy. Nature 549, 203–209 (2017).
[25] Vasileska D, Goodnick S M, Klimeck G. Computational Electronics: semiclassical and quantum device modeling and simulation[M]. CRC press, 2017.
[26] Ajagekar A, Humble T, You F. Quantum computing based hybrid solution strategies for large-scale discrete-continuous optimization problems[J]. Computers & Chemical Engineering, 2020, 132: 106630.
[27] Henry Corrigan-Gibbs, David J. Wu, and Dan Boneh. Quantum Operating Systems[J]. HotOS ' 17: Proceedings of the 16th Workshop on Hot Topics in Operating Systems.
[28] Honan, R., Lewis, T.W., Anderson, S. and Cooke, J., 2020, October. A Quantum Computer Operating System. In International Conference on Algorithms and Architectures for Parallel Processing (pp. 415-431). Springer, Cham. https://www.riverlane.com/news/2020/12/riverlane-release-first-version-of-quantum-operating-system-deltaflow/.
[29] https://www.riverlane.com/news/2020/12/riverlane-release-first-version-of-quantum-operating-system-deltaflow/
[30] E. Lucero, M. Hofheinz, M. Ansmann, R. C. Bialczak, N. Katz, M. Neeley, A. D. O' Connell, H. Wang, A. N. Cleland, and J. M. Martinis. High-fidelity gates in a single Josephson qubit. Phys. Rev. Lett., 100:247001, Jun 2008.
[31] Richard Jozsa (1994) Fidelity for Mixed Quantum States, Journal of Modern Optics, 41:12, 2315-2323.
[32] Leonardo Banchi, Samuel L. Braunstein, and Stefano Pirandola, Quantum

Fidelity for Arbitrary Gaussian States, Phys. Rev. Lett. 115, 260501.
[33] Inverso O, Trubiani C. Parallel and distributed bounded model checking of multi-threaded programs[C]//Proceedings of the 25th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. 2020: 202-216.
[34] Asyabi E, Sharafzadeh E, SanaeeKohroudi S A, et al. CTS: An operating system CPU scheduler to mitigate tail latency for latency-sensitive multi-threaded applications[J]. Journal of Parallel and Distributed Computing, 2019, 133: 232-243.
[35] Chen J, Du C, Xie F, et al. Scheduling non-preemptive tasks with strict periods in multi-core real-time systems[J]. Journal of Systems Architecture, 2018, 90: 72-84.
[36] Hartmut Neven, John Martinis, et al. Physical qubit calibration on a directed acyclic graph. arXiv:1803.03226v1 [quant-ph].
[37] Zhao-Yun Chen, Guo-Ping Guo. QRunes: High-Level Language for Quantum-Classical Hybrid Programming. arXiv:1901.08340[quant-ph].
[38] Andrew S. Tanenbaum and Herbert Bos. 2014. Modern Operating Systems (4th. ed.). Prentice Hall Press, USA.
[39] Jun Kawahara, Toshiki Saitoh, and Ryo Yoshinaka. The time complexity of the token swapping problem and its parallel variants. In International Workshop on Algorithms and Computation, pages 448–459. Springer, 2017.
[40] Gushu Li, Yufei Ding, and Yuan Xie. Tackling the qubit mapping problem for NISQ-era quantum devices. In 24th ASPLOS, pages 1001–1014. ACM, 2019.
[41] Prakash Murali, Jonathan M Baker, Ali Javadi Abhari, Frederic T Chong, and Margaret Martonosi. Noise-adaptive compiler mappings for noisy intermediatescale quantum computers. In 24th ASPLOS, pages 1015–1029, Providence, RI, USA, 2019. ACM.
[42] Robert Wille, Lukas Burgholzer, and Alwin Zulehner. Mapping quantum circuits to IBM QX architectures using the minimal number of SWAP and H operations. In 56th DAC, page 142. ACM, 2019.
[43] Haowei Deng, Yu Zhang, and Quanxi Li. 2020. Codar: a contextual duration-aware qubit mapping for various NISQ devices. In Proceedings of the 57th ACM/EDAC/IEEE Design Automation Conference (DAC ' 20). IEEE Press, Article 208, 1–6.
[44] Bin-Han Lu, Yu-Chun Wu, Wei-Cheng Kong, Qi Zhou, Guo-Ping Guo. Special-Purpose Quantum Processor Design. arXiv:2102.01228 [quant-ph].
[45] Bonnet, É., Miltzow, T. & Rzążewski, P. Complexity of Token Swapping and Its Variants. Algorithmica 80, 2656–2682 (2018).
[46] Kelly, J., Barends, R., Fowler, A. et al. State preservation by repetitive error detection in a superconducting quantum circuit. Nature 519, 66–69 (2015).

Note: Figure translations are in progress. See original paper for figures.