引用格式：刘磊, 王卓浩, 李芃羽. 量子操作系统（QuOS）及软件栈的设计 [J]. 电子科技大学学报, 2025, 54(2): 210-220.
LIU L, WANG Z H, LI P Y. Quantum computing operating system (QuOS) and software stack[J]. Journal of University of Electronic Science and Technology of China, 2025, 54(2): 210-220.

量子操作系统（QuOS）及软件栈的设计

刘 磊*，王卓浩，李芃羽

（北京航空航天大学 计算机学院，北京 100191）

摘要：量子计算机潜力巨大，是国内外积极关注的颠覆性计算技术，已经逐步进入人们的视野，被赋予能够高效解决经典计算机不能解决的问题期望。该文首先总结了国内外量子计算机系统的发展（涉及硬件、软件多个层次），并归纳了目前量子计算系统需要面对的问题以及相关的解决方案和方向。随后，提出了量子操作系统的设计原则，并介绍了一种量子操作系统 QuOS 的原型核心机制。

关键词：量子计算机体系结构；量子操作系统；量子计算；量子计算软件栈
中图分类号：TP316; O413 **文献标志码：**A DOI: 10.12178/1001-0548.2022360

Quantum computing operating system (QuOS) and software stack

LIU Lei*, WANG Zhuohao, and LI Pengyu

(School of Computer Science and Engineering, Beihang University, Beijing 100191, China)

Abstract: Quantum computers hold immense potential and are a disruptive computing technology that is receiving active attention both domestically and internationally. Quantum computers have gradually entered the public's view and are expected to efficiently solve problems that classical computers cannot address. This paper first summarizes the development of quantum computer systems, covering multiple levels of hardware and software. It also summarizes the existing issues in quantum computing systems, as well as the corresponding solutions and directions to be studied. Finally, the paper presents the design principles of a quantum operating system and introduces the core mechanisms of a prototype quantum operating system (QuOS).

Key words: quantum computer architecture; QuOS; quantum computing; quantum computing software stack

我们已处在“后摩尔定律”时代，经典的以 CPU 为主的计算架构已经发生了深刻的变革。加速器、协处理器、大小核等计算架构已经在过去几年里成为主流服务器、移动设备等的关键计算架构，新型计算模式、计算架构不断涌现。量子计算有可能为基础计算架构带来颠覆性变革，是我国科技战略的重要发展方向，同时也是国际上新的科技战略高地。业界普遍认为量子计算能处理某些用经典计算机几乎不可能完成的任务。如中国科学技术大学研发的“九章”量子计算原型机在求解高斯玻色取样任务中展现了量子计算的优越性[1]；同时，量子计算也可被应用于机器学习[2-3]、化学研究[4-5]等领域；一些工作也证明了量子计算与经典计算机相比，在海量信息检索[6]、大整数质因数分解[7]等方面能产生指数级的加速。

量子计算机利用量子位纠缠态和叠加态，极大地提高了计算的并行性。然而，与经典计算机相比它有着截然不同的计算与应用架构。目前，量子计算芯片的物理量子位（Qubit）有超导、离子阱等材质，超导量子芯片需要大功率制冷设备的协助，才能短时间（500～1 000 μs）保持量子一致状态并完成计算[8]。但是，由于目前量子芯片架构及其工艺、量子操作系统、软件栈等要素组件尚处于快速发展阶段，量子计算机尚不能完全展现其巨大

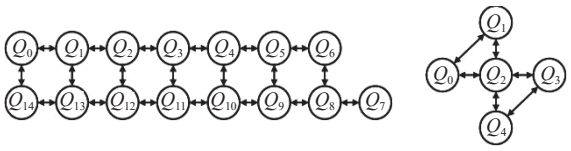
收稿日期：2023−04−10
基金项目：国家重点研发计划（2024YFB4504003）；国家自然科学基金（62072432）
作者简介：刘磊，博士，教授，主要从事计算机体系结构、量子计算机系统等方面的研究。
*通信作者 E-mail: liulei2010@buaa.edu.cn

优势[9]。

1　发展概况

本文从 3 个方面介绍目前量子计算机技术的发展概况和主要问题。

1） 基础计算架构与设备。量子计算机（量子芯片）通常采用超导（如 IBMQ、谷歌悬铃木）、离子阱（UMD）、光量子（“九章”）等实现方式[1,10-11]。但出于对芯片工艺、集成、维护等诸多因素的考量，采用超导器件是较为理想的选择。目前主流的超导量子芯片是 2D 的 Mesh 结构，如图 1 所示[12]。可以看出，每个超导量子位（Qubit）与其附近的量子位产生连接，有链接的量子位之间可进行 CNOT 等量子计算操作。在 2D 结构的量子芯片中，并非所有的量子位之间均有连接，也并非每个量子位向外连接其他量子位的数量都是相同的。并且，每个物理量子位的实时状态均不一样，导致状态的保持时间、运算准确度均不相同，在有些情况下甚至差异很大[13]。此外，量子位之间的连接器也会存在不稳定的现象进而影响计算的准确度（Fidelity）。在计算过程中，如果两个逻辑量子位需要进行 CNOT 等操作，但它们之间的物理位置在芯片上并不相邻，则需要通过 SWAP 交换操作将这两个逻辑量子位移动到有连接的直接相邻（Neighbour）的位置。SWAP 操作依赖于量子位之间的连接器，每次 SWAP 操作可能会影响量子位的状态，SWAP 操作越多，则越有可能导致最终的计算结果产生偏差。由于这些问题，量子线路/算法在芯片上运行时常会产生不稳定的运算结果，运算的准确率时常波动。



a. IBMQ_melbourne　　b. IBMQ_yorktown

图 1　IBMQ_melbourne 和 IBMQ_yorktown 的 2D Mesh 结构

以 IBM_washington[14] 量子芯片为例，其拥有 127 个物理量子位。在该芯片中，量子位一致状态保持时间平均为 101.6 μs，而执行单个量子门操作所需的平均时间为 550.41 ns。因此，可在该量子芯片上执行的量子线路（程序）的深度将会受限。理论上量子线路深度超过约 180 后，就难以保证量子计算结果的准确性。此外，量子芯片中量子位、连接器等部件也存在操作错误率且不尽相同，时刻影响着计算结果和效果。如在 IBM_washington 中 CNOT 门操作错误率最低可达 0.59%，最高可达 17.3%，平均为 1.92%；量子位读出错误率最低可达 0.30%，最高可达 35.60%，平均为 3.13%。综上所述，量子计算基础架构处在不断发展进步的阶段，如何提高量子计算芯片等基础计算架构的可用性、稳定度、可靠度及其准确率是需要解决的问题。

2）量子操作系统及软件栈。操作系统（OS）及软件栈是计算机系统的“魂”，可用性强的计算机系统不可能脱离系统软件体系的支撑而独立存在，量子计算机也如此。

目前，量子计算机的存在形式多种多样，包括量子设备型与量子芯片型等。典型的设备型量子计算机类似于“九章”光量子设备[1]，其是一台用以完成特定任务的专用的光量子计算系统。典型的基于量子芯片的量子计算机系统（如 IBMQ、谷歌悬铃木等）通常具备一定的可编程性，量子位（采用固态超导材质较多）的排布呈现 2D/3D 的 Mesh 结构，目前可集成 100 个量子位左右（如 IBMQ 最新的量子芯片 IBM_washington 可集成 127 个物理量子位[14]，谷歌的 Bristlecone 量子芯片集成了 72 个物理量子位[15]）。量子操作系统及系统软件栈向用户隐藏了量子计算机底层的具体物理实施细节，为上层量子算法、量子程序的执行提供接口。当执行一个量子程序时，用户将高级语言（如 QASM[16]，Scaffold[17]）表示的量子算法作为输入，经过系统软件栈的转换、优化、调度和映射，最终生成能够在特定量子计算机硬件上执行的机器指令序列。

“量子程序/线路的映射”是量子软件栈及 OS 的关键挑战之一。量子程序/线路需要被映射到量子位上才能运行。映射机制关系到量子程序能否运行在有较高稳定性、较高准确性的物理量子位上；也关系到昂贵的量子芯片的利用率、可用度和运行准确率。随着量子芯片的发展，量子程序映射机制也进行着深刻的变化。如先前的映射机制仅将量子芯片抽象为 1D 或 2D 简化模型[18-19]，后续引入了针对实际芯片拓扑结构的考量[20-21]。对量子门操作错误率的优化[13]、对串扰错误的考虑[22-23] 也逐步被加入到量子程序映射机制中。近期，随着量子计算云服务的发展，提升量子计算机的通量和资源利用率成为了一项重要需求；文献 [24-26] 提出了同时映射多个并发量子程序的解决方案。此外，并发调度多个量子程序也能够帮助加速量子机器学习领域中

的变分量子算法（VQA）[27] 等。

并发量子程序映射问题又对现有的量子程序映射机制提出挑战。现有的映射策略主要支持单程序，难以在两个量子程序间进行公平的资源分配，难以保证每个量子程序的执行保真度。研究者普遍认为，随着量子计算机架构的快速发展，量子程序映射策略将不断更迭，以充分挖掘量子计算机硬件的能力，保证计算的可靠度、准确度。

3） 量子算法/线路的优化。量子算法的实现媒介是量子线路；量子线路通过各种量子门操作物理量子位实现量子算法进而达到计算目的。图 2 展示了 Grover 算法[6] 对应的量子线路，其功能是在 N 个无序数据中搜索一个特定数据。量子线路是量子算法的显化，决定着在量子芯片上如何进行计算。因此，优化量子线路就是优化量子计算的过程。

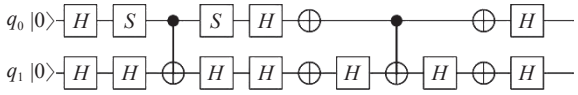


图 2 Grover 算法对应的量子线路图（搜索“01”）

从物理机的角度来讲，目前能够集成的量子位数量还比较有限。IBM 称已制造出的量子计算机中最多包含 1 121 个物理量子位。然而由于量子位数目和错误率的限制，目前的量子计算机仍不能运行大规模的量子程序（如破解 RSA-2048 预计需要 4 000 个以上量子位）。因此，如何优化量子算法，优化（减小）量子线路的规模，使其可在现有的量子计算机上成功运行并准确输出结果，就成为至关重要的研究内容。被优化的量子线路需要与被优化之前保持相同的准确度（Fidelity），否则优化就失去了意义。酉矩阵的分解[28-29] 是量子算法优化关注的一个重点问题。每个量子算法可以表示为一个酉矩阵。在量子计算机上执行量子算法时，需要将酉矩阵分解为目标量子计算机支持的基础门操作的组合。优化的目标是减少酉矩阵分解后的量子位占用和量子线路深度。另一个问题是量子线路的转化[30-32]，主要研究如何通过等价转换优化现有量子程序，消除量子线路中的冗余操作，进一步降低量子线路的规模和深度。

由于当前量子计算机尚不普及，利用超级计算机集群模拟大规模量子线路运行已成为一种普遍方法[33-34]，也是国内外研究的热点。从这一角度来看，对量子算法、线路的优化也能够降低超算模拟的复杂性和运算成本。

此外，业界还积极探索经典计算机与量子计算机的协同工作方式。首先，量子计算机可作为经典计算机的协处理器。现代计算机通常采用 CPU 与 GPU、FPGA 等异构加速器或协处理器协同完成特定计算任务。文献 [35] 提出可将量子芯片作为一种“qFPGA”协处理器，专门加速适合量子计算的任务。其次，还可结合量子计算机与经典计算机的计算能力，协同完成复杂量子计算任务。如将大型量子线路拆分为多个小规模的部分[36-40]，分别由多个量子计算机或经典计算机执行，最后合并结果。这种方法虽然允许协同量子计算机和经典计算机执行更大规模的量子程序，但也面临诸多挑战，如灵活性不足和量子线路拆分导致计算开销高，仍存在较大的优化空间。

本文认为，以上 3 个层次之间存在深刻关联，彼此互相依赖，是共同构成量子计算机系统实用化的要素。量子计算机系统不仅包括底层计算架构，还应还涵盖完整的软件栈及优化机制。目前，国内外相关科研机构正在积极部署相关研究与技术攻关，量子计算系统尚处在蓬勃发展阶段。在量子芯片方面，IBM、谷歌等公司努力提升单个量子芯片的比特集成数量，但现阶段超导量子芯片的量子比特数量仍在 100 个左右[14-15]，难以满足许多实用量子算法的需求。在量子计算软件栈方面，国内外正处于技术快速发展时期，创新和优化仍有很大空间，如并行量子任务调度、量子算法纠错及任务在量子芯片上的映射等。在量子算法/线路优化方面，已有诸多研究致力于探索利用量子算法解决实际问题[4-7]，同时也有尝试将人工智能与量子计算相结合[2-3]。

2 国内外代表性的相关工作

为推进量子计算的实用化，国内外学术界、工业界从量子芯片架构及工艺[10-11, 41-51]、量子操作系统及软件栈[13, 18-27, 35, 47, 52-72]、量子算法优化[28-32] 方面开展了积极的探索。现有工作探索了如何通过改进硬件技术工艺、优化系统软件栈的方式提升量子程序的保真度，或更高效可靠的运行更大规模的量子程序。

针对量子芯片架构及工艺优化，作出了一系列积极的探索工作，内容涉及量子位及连接器制造工艺[10-11, 41-45]、量子内存技术[46-47]、量子芯片集成和封装[48-49] 等关键技术。有研究者提出了针对特定量子线路的量子芯片设计流程[50-51]，根据该类量子线路

的特点，确定物理量子位的放置位置、连接和频率，设计专用量子芯片。这些技术提升了量子芯片的可靠性，但要使量子芯片进一步向实用化迈进，需要量子操作系统和系统软件栈的跨层次协同设计。

为优化量子程序准确度，研究者针对量子系统软件栈中的各个层次进行了改进和优化。其中包括量子程序映射、错误抑制、并发程序调度、调试与断言、量子纠错编码等。量子程序映射问题已被证明是 NP-完全问题[20, 52]，求解该问题的技术可分为两类。一类方法[20, 22, 52, 55-59]将量子程序映射问题转换为等价的有约束条件的优化问题，并采用求解器[53-54]求解。此类方法可获得小型量子程序映射问题的最优解或近似最优解，但算法时间复杂度高，难以支持求解具有数十个以上量子位的量子程序映射问题。第二类方法[13, 18-21, 57, 60-64]采用启发式信息辅助搜索，具有更优的可扩展性，不受限于量子芯片和量子程序的规模，具有处理更大规模量子程序映射问题的能力，但无法保证所求解结果为最优解。一部分工作[18-19, 60-61]对量子芯片结构进行了简化，针对 1D 线性模型或 2D 网格模型设计了启发式算法。另外的工作[13, 20, 21, 57, 62-64]则基于具体的芯片拓扑结构。此外，为提升量子程序的执行保真度，设法对量子芯片上易发生的错误进行抑制，包括串扰错误抑制[22-23, 65]、测量错误抑制[66]，以及通过采用多种初始映射对局部不可靠资源错误的抑制[67]等。文献 [13, 25-27] 对并发量子程序映射问题进行了探索，支持在同一量子芯片上同时映射和执行多个量子程序。文献 [25-26] 提出了并发量子程序调度机制，允许依据用户需求，自动判断是否采用并发机制，并能够选择最优的并发量子程序共同执行。这些工作有助于量子计算机在量子计算云服务逐渐兴起的背景下提升资源的利用率和通量，也能帮助加速变分量子算法（VQA）的执行。文献 [68-69] 设计和改进了量子程序中的断言机制，通过在量子线路中设置断言，检测量子位是否满足预期状态，协助完成量子程序调试。量子纠错编码（QEC）能检测并纠正量子程序执行期间发生的错误，是未来实现容错量子计算的重要保证。文献 [47,70-72] 对量子纠错编码的实现和优化进行了探究，对构造一个纠错码所需的物理量子位数目、量子纠错码的执行效率进行了优化和改进。

量子程序/线路的优化可消除量子线路中的冗余操作，在保证功能及运算结果不变的情况下简化量子线路的复杂度。文献 [30,73] 是研究量子程序优化的代表性工作之一，核心方法是利用冗余量子位缩减量子线路深度，从而缩短量子程序执行时间，避免由于执行时间过长导致不能保持一致状态。

目前用于求解大规模的量子计算机还处在发展阶段，因此使用经典计算机搭建量子计算模拟系统，仍是目前研究量子算法和验证量子计算有效性的必要方式。国内外许多机构设计并开发了量子计算模拟器。如阿里巴巴发布的“太章”模拟器[74]、英特尔的量子模拟器“Qhipster”[75]、牛津大学的通用量子模拟器“QuEST”[76]等。这些模拟器各有其应用领域，在模拟效率、规模、扩展性等方面也各有优势。使用经典计算机模拟量子计算的难点在于随着模拟的量子位数的增长，模拟的时间和空间开销呈指数级增长[74]，对于大规模量子计算的模拟单机已无法满足。文献 [74, 77-80] 基于大规模集群模拟量子计算，通常采用任务拆分[36-40, 74, 77-78]、节点数据传输优化[79-80]的方式优化超算集群的并行处理能力。然而目前模拟器能模拟的量子位数、量子应用仍然有限，且消耗的资源较大，模拟效率也不高。量子模拟器的整体效能有待进一步优化。此外，混合计算架构（经典计算机与量子计算机协同工作）是运行更大规模量子程序的另外一种可行方法。文献 [36-40] 提出可将较大规模的量子线路拆成多个小规模的线路，使之在多台小型量子计算机上分别执行或在经典计算机上模拟执行，最终再将运算结果汇总，以达到处理较大规模量子线路的目的。

3　量子操作系统（QuOS）及软件栈

3.1　量子操作系统的设计原则和操作系统行为

3.1.1　资源管理与分配

对于超导量子计算机来说，量子位的资源分配就是量子操作系统的关键行为。在单个量子计算节点上，QuOS 为量子程序分配量子位，也能够为并发的量子程序分配量子位。具体包括以下 5 个方面。

1）单个量子程序的资源分配：QuOS 依据量子芯片的实时状态，通过综合评估量子位的稳定性和错误率，优先选择稳定性高且错误率低的量子位及连接进行分配，从而提升量子程序的准确度，并确保量子计算机的可靠度。

2）并发多量子程序的调度：QuOS 能够综合量子芯片的实时运行状况和待调度量子程序的资源需求，选择最优的并发量子程序组合，在最大化并发程序的总体准确性的同时，提升量子计算机这一

昂贵资源的利用率，为构建高可靠性的“量子云”服务提供技术支持。

3）程序隔离机制：在多量子程序并发的场景下，QuOS 通过程序隔离技术降低串扰（Crosstalk）等物理因素带来的影响，同时隔离运行异常的程序，以保障其他程序的正常运行。

4）量子位交换（SWAP）支持：对于单量子程序，QuOS 支持高效且路径最短的量子位交换操作，以减少由于交换导致的准确度损失。在多程序并发的场景下，QuOS 支持跨程序量子位交换操作，以提高整体资源调度的灵活性。

5）多任务调度能力：QuOS 支持单量子任务和多量子任务的调度。具体而言，任务分为两类：①完全在量子计算机上执行的任务；②一部分在量子计算机上运行，一部分在经典超算上运行的混合计算任务。

3.1.2 掩蔽限制

QuOS 对上层用户隐藏硬件体系结构层次的细节。目前看来，量子计算的计算架构呈现以下 3 种计算形态：1）将经典计算机上任务的一部分划分到量子计算机上执行，以加速计算过程，从而提升整体任务的执行效率；2）将量子计算任务的一部分交由经典超算进行模拟，以克服目前因量子位数量不足而导致的计算瓶颈；3）将量子计算任务拆分为多个子任务，并分配到不同的量子计算机上执行，实现“任务－计算资源”的最佳匹配，从而更高效的利用量子计算资源。

为实现对用户隐藏这些底层异构混合计算架构细节的目标，QuOS 包含面向异构量子计算的调度机制，这一机制能够将不同类型的任务映射到合适的量子计算机上，并支持基于用户标注的任务特性和量子硬件的实时状态信息进行任务调度。经过异构计算架构中不同处理单元处理的结果最终会被融合，并汇总成一个统一的运算结果。在此过程中，分批执行的中间结果必须被临时存储并维持一段时间，以等待其他程序片段的结果完成后进行合并，但量子态无法长时间保持稳定，这是一对必须要被解决好的、天然存在的矛盾。为解决此问题，QuOS 引入了一种基于“谐振腔”的量子数据存储架构及量子态存储/读取操作方法，该方法将量子线路的中间处理结果保存在腔体中。按目前的技术指标，这种存储方法能够使量子态的保持时间至少提高两个数量级。需要操作时，存储的量子态可从谐振腔中读出，进行后续处理，并确保在操作过程中保持量子态的硬实时性，避免量子状态的变化。

3.1.3 统一服务

QuOS 为用户提供统一的服务接口，具体功能包括以下 3 个方面。

1）统一 API 提供编程抽象：QuOS 提供统一的 API，便于程序员进行高效的编程抽象，程序员可以在编写程序时显式的标注关键语义，如程序中哪些部分将在量子计算机上执行，异构计算部件之间如何协同等。在程序运行时，QuOS 将根据这些标注的信息在不同设备之间调度任务。

2）量子线路模型固化：考虑到目前量子计算机的“专用性”，QuOS 可预先在系统中固化一些能够解决经典计算问题的量子线路模型（如 oracle 算子），以降低提供统一服务的开销。在计算过程中，只要待解决的问题确定了，就可以在量子计算机上启动相应的量子线路。基于对问题的抽象，经典计算机将输入信号传递给量子计算机，通知量子计算机开始处理。量子计算机处理完成之后，经典计算机将接收处理结果。此计算模型特别适用于某些检索类问题以及在大范围空间中寻找最优解的问题。同时，借助量子计算机的加速，架构师和程序员在系统设计或编码时也可以更大胆地探索以往传统计算机无法或不敢处理的复杂问题。

3）量子设备间的协同工作：QuOS 需要协同多种量子计算设备的工作，并确保它们之间的通信。对于上层用户而言，无需了解底层计算设备的详细信息，QuOS 应当屏蔽大多数量子计算机运算的细节。信息传递由 QuOS 层通过管道（QuPipe）或软总线实现，确保不同设备间的数据流畅传输。

3.1.4 稳定健壮

量子线路必须在极短的时间内执行完毕（通常不超过数百微秒），并且需要多次重复执行并观测结果出现的概率。为了保证量子计算的可靠性，QuOS 需要支持针对量子线路特点的调试、断言、纠错算法以及结果验证等机制；此外，QuOS 还需确保量子线路在运行过程中的稳定性，保证输出结果的可推演性，并力求在微秒级时间内修复损坏的量子线路。

3.2 QuOS 的核心设计[25-26]

本文提出了一种基于超导量子计算机[25-26]的量子操作系统（QuOS）的原型系统，其中包括若干核心机制。量子操作系统作为量子计算机系统资源的管理者，负责映射和调度量子程序/任务，并分配和管理量子计算机硬件资源。本文介绍了面向超

导量子位 NISQ 量子计算机（如 IBMQ）设计的操作系统原型。

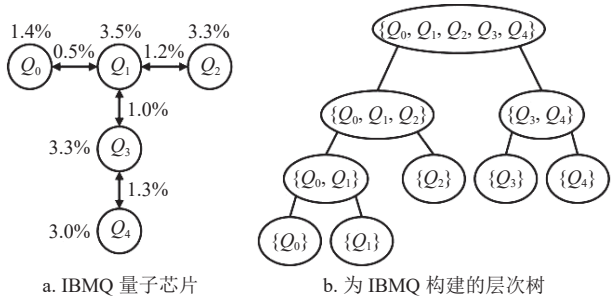
3.2.1　量子程序映射机制及优化

量子程序或量子线路需要被映射到物理量子计算机上才能执行，而管理量子程序映射是 QuOS 的核心职能。量子位的映射结果直接决定着量子程序的运行效果和输出结果。针对最新 NISQ 量子计算机的特性，QuOS 设计了量子程序映射框架，用以提升量子程序的准确度、增强量子计算机的可靠性和效率。具体包括以下 4 项关键技术。

1）高内聚的量子位映射

量子位映射时结合物理量子位的实时状况，采用“社区发现”等算法对健壮量子位资源进行社区划分，将物理连接紧密且可靠的健壮量子位划分至同一社区中，搜索最优量子位区域进行分配，提升分配资源可靠度且避免可靠量子位资源的浪费。其总体思路是：根据量子芯片中量子位的物理拓扑，以及门操作、读出和串扰错误率，构建一棵层次树，对量子芯片中的健壮资源分布进行建模，帮助定位量子芯片上连接紧密且可靠的健壮资源，从而为量子程序提供更优的初始映射。层次树的构建依据量子位的物理拓扑和错误率，以保证层次树中每个节点内量子位的可靠性和内聚性。

具体而言，QuOS 采用基于“社区发现”的过程构造一棵二叉层次树。其中每个节点代表一个社区或候选的待分配量子位区域。初始时，每个物理量子位对应层次树中的一个叶结点。在每次迭代中，QuOS 选取 2 个能使收益最大化的层次树节点进行合并，并将这 2 个节点作为合并后新节点的左右子节点，直到最终只剩下一个包含所有物理量子位的根节点为止。图 3 展示了对于 IBMQ London 量子芯片构建的层次树。其中图 3a 为 IBMQ 量子芯片，每个节点对应的数值代表该物理量子位的读出操作错误率，连接上的数值代表两个物理量子位间的 CNOT 操作错误率。图 3b 则展示了依据“社区发现”方法为 IBMQ 构建的层次树[25-26]。



a. IBMQ 量子芯片　　b. 为 IBMQ 构建的层次树

图 3　IBMQ London 量子芯片层次树

层次树节点合并时的收益判断标准，主要包括两项。第一项关注社区中量子位连接的紧密程度，具体通过比较两个社区合并前后社区划分方案的模块度差异来衡量，模块度用于衡量一种社区划分方案的合理程度。差值越大，说明合并两个社区后，每个社区内部量子位连接更紧密。第二项关注社区内量子位和连接器的可靠度。具体通过量子芯片中的门操作错误、串扰（Crosstalk）错误和读出错误进行衡量。QuOS 倾向于优先合并具有低读出、操作、串扰错误率且连接紧密的量子位。因此，量子位集合越健壮，越容易被优先合并，其在层次树中的深度就越深。并且一个社区中的物理量子位健壮且具有紧密的相互连接，能够帮助降低量子程序映射成本。

2）并发多量子程序的映射

为提高量子计算机的通量和资源利用率，QuOS 支持并发量子程序任务映射机制，即在一个量子芯片上同时映射多道量子程序。

QuOS 基于前文所述的层次树，为每个并发量子程序分配初始映射区域，以支持并发量子程序映射。在每个量子程序的映射过程中，系统自底向上搜索层次树，可搜索到多个可供分配的量子位集合。最终依据以下 3 项原则，选取最优的量子位集合进行分配。

首先，候选量子位集合中的量子位连接越紧密，越有助于降低量子位调度（SWAP）成本。对于一个量子位集合，其量子位连接的紧密程度可由其中每对量子位的 SWAP 路径长度（即最短路径−1）的平均值进行衡量。优先选取平均 SWAP 路径长度更小的量子位集合，以降低后续量子位调度的 SWAP 开销。其次，候选量子位集合中门操作和读出错误率越低，越有助于提升量子程序的执行可靠度。在超导量子芯片中，CNOT 操作错误率和读出错误率是导致错误的主要因素。因此选取具有更低平均 CNOT 操作错误率和读出操作错误率的量子位集合，以获得更高的程序执行保真度。最后，应尽量避免将易发生串扰问题的量子位分配给并发量子程序，以缓解串扰错误。由于设备制造缺陷、控制错误等原因，在邻近的一对量子位连接上同时执行的 CNOT 操作可能发生串扰，导致两个 CNOT 操作的错误率增加。在社区分配的过程中，为了避免串扰错误，考虑易发生串扰的 CNOT 对的数量多少比考虑串扰错误的高低更为重要。其原因在于：即使一对 CNOT 操作的串扰错误较高，也可通过指令调度等技术避免串扰的发生。然而，

当易发生串扰错误的 CNOT 对数过多，需要大量的指令调度操作来避免串扰，这会导致被延迟执行的 CNOT 操作增多，从而增加量子程序的执行时间，带来与量子位状态保持时间相关的错误。因此，本文优先选取与已分配的社区易串扰 CNOT 对较少的量子位区域进行分配。

3）跨程序的量子位交换（SWAP）调度

QuOS 针对并发量子程序映射特点，引入跨程序 SWAP 操作，即逻辑量子位的交换可以在程序之间进行，而不仅限于单个程序内部，进而降低 SWAP 操作的成本，提升程序可靠度。

文献 [25-26] 证明，在以下两种情况下执行跨程序 SWAP 操作，相比仅允许执行单个程序内部 SWAP 操作，能够降低量子位调度的开销。

情况①，如图 4 所示。先前策略对每个程序分别进行量子位调度，不允许跨程序 SWAP 操作。而 QuOS 允许对多个量子程序同时进行量子位调度，并引入跨程序 SWAP 操作，使得 1 个跨程序 SWAP 操作可替代 2 个程序内 SWAP 操作[25-26,81]。

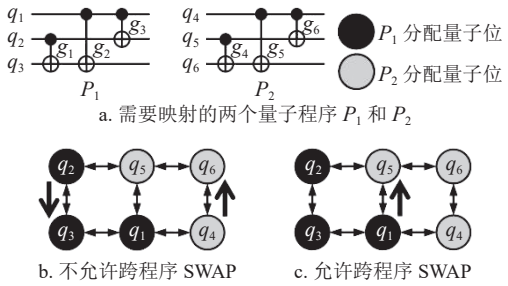


图 4 跨程序 SWAP 操作可替换 2 个程序内 SWAP 操作

情况②，如图 5 所示。先前策略中的最短 SWAP 路径可能被已映射的量子程序遮挡，需 3 步 SWAP 操作才能使量子位 q_1 和 q_5 相邻。而 QuOS 采用跨程序 SWAP 操作，可采取捷径，仅需 1 步 SWAP 即可达到相同目的[25-26, 81]。

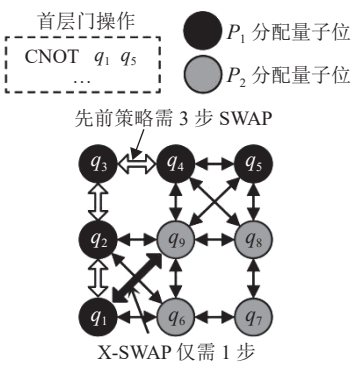


图 5 跨程序 SWAP 可取捷径

QuOS 提出一种新的量子位调度机制，能够同时调度多个并发量子程序，并充分利用跨程序 SWAP 操作，降低量子位调度的开销，具体包括以下 3 项内容。

①启发式搜索空间设计。本文以首层 CNOT 门操作是否存在紧邻的后继 CNOT 为判断依据，其中关键门操作为存在紧邻后继的 CNOT 门操作，筛选首层 CNOT 门操作中的关键门操作。并优先处理关键门操作，以便快速更新首层 CNOT 门操作，缩小映射后的量子线路深度。因此，启发式搜索空间限定为仅涉及关键门操作的物理量子位相关的 SWAP 操作。

②启发式函数设计。最近邻成本函数（NNC）通常可作为启发式函数的一部分，它代表门操作集合中所有 CNOT 操作的平均 SWAP 路径长度，可帮助选取最少的 SWAP 操作完成量子位调度。除此之外，QuOS 把允许和不允许跨程序 SWAP 两种情况下的最短 SWAP 路径长度差值加入启发式函数，鼓励执行能够采取捷径的跨程序 SWAP 操作。

③QuOS 考虑了串扰错误的影响。本文分析表明，引发串扰错误的主要原因在于同时执行的 CNOT 门操作和量子位调度过程中引入的 SWAP 操作。为了缓解串扰错误，QuOS 改进量子位调度的启发式搜索过程，每次迭代时，不再只选取具有最低（最优）启发式函数值的 SWAP，而是依据启发式函数值由低到高选出一组较优的 SWAP，再从中选择串扰错误最低的 SWAP 操作。

4）针对新 3D 架构量子芯片的映射。对于新型 3D 量子芯片，可充分利用量子位的三维排布和更多的量子位间相互连接，优化初始分配和 SWAP 策略，从而缩短 SWAP 路径并提升程序执行保真度。QuOS 利用具有高度数的物理量子位，对 3D 量子芯片上量子程序的初始分配进行优化。对于量子程序中的一个逻辑量子位，如果与之交互的逻辑量子位数目较多，则将其分配至一个具有较高度数的物理量子位上，可有效降低 SWAP 成本。这一策略会使更多 CNOT 操作所涉及的逻辑量子位在物理量子位上的映射位置直接相邻，从而避免在执行前插入 SWAP，降低量子位调度过程中的 SWAP 成本。

3.2.2 多量子程序并行的调度

针对不同量子芯片的特性及多样化的量子程序，QuOS 提供了一种自适应的量子任务调度机制，能够自动控制任务单独执行或并发执行多任

务。该机制通过筛选可并发执行的最优量子程序组合，在提升量子计算机通量和量子位资源利用率的同时，避免量子程序执行准确度的大幅下降。

QuOS 采用了一种按需调度的自适应量子程序调度机制，支持根据系统或用户需求，选择最优量子程序组合执行。该技术方案主要包括以下两项内容。

1）调度器的优化目标选择。QuOS 根据用户的输入或自动检测系统当前的任务，以提升通量或提升量子程序执行保真度为目标。如 VQA 量子算法包括数百个串行、重复且独立的执行和测量过程，每个独立的执行和测量过程可当做一个单独的量子程序进行调度。此时，如果调度器的优化目标是提升通量，则可加速 VQA 算法执行。而在需要保证一个量子程序的保真度时，调度器可以单独映射该量子程序，避免并行其他程序对其执行精度造成干扰。

2）调度器的调度逻辑。调度器首先选取调度队列中的首个量子程序作为待执行任务。随后按照与首个量子程序的匹配度顺序，检查调度队列中是否存在其他能够与该量子程序并发执行，且造成的成功率的损失低于阈值的程序。若存在此类程序，则将这些任务与该量子程序同时映射到量子计算机上执行；否则，单独映射并执行该量子程序。通常，与该量子程序深度相近且所需量子位数较少的程序，具有较高的匹配度，适合并行执行。

4　结束语

本文总结了量子计算系统的发展，提出了一种量子操作系统（QuOS）及软件栈的设计方案。虽然量子计算机和经典的冯诺依曼计算机架构有着本质的区别，但是业界关于经典计算机操作系统的研究已经有了丰富的经验和丰厚的积淀。“他山之石，可以攻玉”，本文认为经典操作系统的设计经验有些同样也适用于对量子操作系统的设计，基于这一思路，提出了相关的设计尝试和改进。希望未来有更多的科研工作者投入到相关的研究中。

参 考 文 献

[1] ZHONG H S, WANG H, DENG Y H, et al. Quantum computational advantage using photons[J]. Science, 2020, 370(6523): 1460-1463.

[2] BIAMONTE J, WITTEK P, PANCOTTI N, et al. Quantum machine learning[J]. Nature, 2017, 549(7671): 195-202.

[3] LLOYD S, MOHSENI M, REBENTROST P. Quantum algorithms for supervised and unsupervised machine learning[EB/OL]. (2013-11-4). https://arxiv.org/pdf/1307.041.

[4] KANDALA A, MEZZACAPO A, TEMME K, et al. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets[J]. Nature, 2017, 549(7671): 242-246.

[5] PERUZZO A, MCCLEAN J, SHADBOLT P, et al. A variational eigenvalue solver on a photonic quantum processor[J]. Nature Communications, 2014, 5: 4213.

[6] GROVER L K. A fast quantum mechanical algorithm for database search[C]//Proceedings of the 28th Annual ACM Symposium on Theory of Computing - STOC '96. New York: ACM, 1996: 212-219.

[7] SHOR P W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer[J]. SIAM Journal on Computing, 1997, 26(5): 1484-1509.

[8] METER R V, OSKIN M. Architectural implications of quantum computing technologies[J]. ACM Journal on Emerging Technologies in Computing Systems, 2006, 2(1): 31-63.

[9] PRESKILL J. Quantum computing in the NISQ era and beyond[J]. Quantum, 2018, 2: 79.

[10] KOCH J, YU T M, GAMBETTA J, et al. Charge-insensitive qubit design derived from the Cooper pair box[J]. Physical Review A, 2007, 76(4): 042319.

[11] DEBNATH S, LINKE N M, FIGGATT C, et al. Demonstration of a small programmable quantum computer with atomic qubits[J]. Nature, 2016, 536(7614): 63-66.

[12] IBM. IBM quantum services[EB/OL]. [2022-2-24]. https://quantum-computing.ibm.com/services?services=systems.

[13] TANNU S S, QURESHI M K. Not all qubits are created equal: A case for variability-aware policies for NISQ-era quantum computers[C]//Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2019: 987-999.

[14] IBM. IBM unveils breakthrough 127-qubit quantum processor[EB/OL]. [2022-2-24]. https://newsroom.ibm.com/2021-11-16-IBM-Unveils-Breakthrough-127-Qubit-Quantum-Processor.

[15] KELLY J. A preview of Bristlecone, Google’s new quantum processor[EB/OL]. (2018-03-05). https://research.google/blog/a-preview-of-bristlecone-googles-new-quantum-processor.

[16] CROSS A W, BISHOP L, SMOLIN J, et al. Open quantum assembly language[EB/OL]. (2017-06-13). https://arxiv.org/pdf/1707.03429.pdf.

[17] HECKEY J, PATIL S, JAVADIABHARI A, et al. Compiler management of communication and parallelism for quantum computation[C]//Proceedings of the 20th International Conference on Architectural Support for Programming Languages and Operating Systems. New

York: ACM, 2015: 445-456.
[18] SHAFAEI A, SAEEDI M, PEDRAM M. Optimization of quantum circuits for interaction distance in linear nearest neighbor architectures[C]//Proceedings of the 50th Annual Design Automation Conference. New York: ACM, 2013: 1-6.
[19] WILLE R, KESZOCZE O, WALTER M, et al. Look-ahead schemes for nearest neighbor optimization of 1D and 2D quantum circuits[C]//Proceedings of the 21st Asia and South Pacific Design Automation Conference. New York: IEEE, 2016: 292-297.
[20] SIRAICHI M Y, DOS SANTOS V F, COLLANGE S, et al. Qubit allocation[C]//Proceedings of the 2018 International Symposium on Code Generation and Optimization - CGO 2018. New York: ACM, 2018: 113-125.
[21] ZULEHNER A, PALER A, WILLE R. An efficient methodology for mapping quantum circuits to the IBM QX architectures[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2019, 38(7): 1226-1236.
[22] MURALI P, MCKAY D C, MARTONOSI M, et al. Software mitigation of crosstalk on noisy intermediate-scale quantum computers[C]//Proceedings of the 25th International Conference on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2020: 1001-1016.
[23] XIE L, ZHAI J D, ZHENG W M. Mitigating crosstalk in quantum computers through commutativity-based instruction reordering[C]//Proceedings of the 58th ACM/IEEE Design Automation Conference. New York: IEEE, 2021: 445-450.
[24] DAS P, TANNU S S, NAIR P J, et al. A case for multi-programming quantum computers[C]//Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture. New York: ACM, 2019: 291-303.
[25] DOU X L, LIU L, DOU X L, et al. A new qubits mapping mechanism for multi-programming quantum computing[C]//Proceedings of the ACM International Conference on Parallel Architectures and Compilation Techniques. New York: ACM, 2020: 349-350.
[26] LIU L, DOU X L. QuCloud: A new qubit mapping mechanism for multi-programming quantum computing in cloud environment[C]//Proceedings of the IEEE International Symposium on High-Performance Computer Architecture. New York: IEEE, 2021: 167-178.
[27] RESCH S, GUTIERREZ A, HUH J S, et al. Accelerating variational quantum algorithms using circuit concurrency[EB/OL]. [2022-3-15]. https://www.xueshufan.com/publication/3197499824.
[28] BARENCO A, BENNETT C H, CLEVE R, et al. Elementary gates for quantum computation[J]. Physical Review A, Atomic, Molecular, and Optical Physics, 1995, 52(5): 3457-3467.
[29] VARTIAINEN J J, MÖTTÖNEN M, SALOMAA M M. Efficient decomposition of quantum gates[J]. Physical Review Letters, 2004, 92(17): 177902.
[30] JIANG J Q, SUN X M, TENG S H, et al. Optimal space-depth trade-off of CNOT circuits in quantum logic synthesis[C]//Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms. New York: ACM, 2020: 213-229.
[31] NASH B, GHEORGHIU V, MOSCA M. Quantum circuit optimizations for NISQ architectures[J]. Quantum Science and Technology, 2020, 5(2): 025010.
[32] WU B J, HE X Y, YANG S, et al. Optimization of CNOT circuits on limited connectivity architecture[EB/OL]. [2022-9-11]. http://arxiv.org/abs/1910.14478v4.
[33] 喻志超, 李扬中, 刘磊, 等. 量子计算模拟及优化方法综述[J]. 计算机工程, 2022, 48(1): 1-11.
YU Z C, LI Y Z, LIU L, et al. Survey of quantum computing simulation and optimization methods[J]. Computer Engineering, 2022, 48(1): 1-11.
[34] DE RAEDT K, MICHIELSEN K, DE RAEDT H, et al. Massively parallel quantum computer simulator[J]. Computer Physics Communications, 2007, 176(2): 121-136.
[35] CORRIGAN-GIBBS H, WU D J, BONEH D. Quantum operating systems[C]//Proceedings of the 16th Workshop on Hot Topics in Operating Systems. New York: ACM, 2017: 76-81.
[36] TANG W, TOMESH T, SUCHARA M, et al. CutQC: Using small Quantum computers for large Quantum circuit evaluations[C]//Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2021: 473-486.
[37] SUCHARA M, ALEXEEV Y, CHONG F, et al. Hybrid quantum-classical computing architectures[C]//Proceedings of the 3rd International Workshop on Post-Moore Era Supercomputing, 2018. DOI:10.13140/RG.2.2.23841.84321.
[38] AYRAL T, LE RÉGENT F M, SALEEM Z, et al. Quantum divide and compute: Hardware demonstrations and noisy simulations[C]//Proceedings of the IEEE Computer Society Annual Symposium on VLSI. New York: IEEE, 2020: 138-140.
[39] PENG T Y, HARROW A W, OZOLS M, et al. Simulating large quantum circuits on a small quantum computer[J]. Physical Review Letters, 2020, 125(15): 150504.
[40] BRAVYI S, SMITH G, SMOLIN J A. Trading classical and quantum computational resources[J]. Physical Review X, 2016, 6(2): 021043.
[41] BARENDS R, KELLY J, MEGRANT A, et al. Coherent Josephson qubit suitable for scalable quantum integrated circuits[J]. Physical Review Letters, 2013, 111(8): 080502.
[42] RIGETTI C, DEVORET M. Fully microwave-tunable universal gates in superconducting qubits with linear couplings and fixed transition frequencies[J]. Physical Review B, 2010, 81(13): 134507.

[43] FRIIS N, MARTY O, MAIER C, et al. Observation of entangled states of a fully controlled 20-qubit system[J]. Physical Review X, 2018, 8(2): 021012.
[44] MAURAND R, JEHL X, KOTEKAR-PATIL D, et al. A CMOS silicon spin qubit[J]. Nature Communications, 2016, 7: 13575.
[45] HAYES A J F, GILCHRIST A, MYERS C R, et al. Utilizing encoding in scalable linear optics quantum computing[J]. Journal of Optics B: Quantum and Semiclassical Optics, 2004, 6(12): 533-541.
[46] NAIK R K, LEUNG N, CHAKRAM S, et al. Random access quantum information processors using multimode circuit quantum electrodynamics[J]. Nature Communications, 2017, 8(1): 1904.
[47] DUCKERING C, BAKER J M, SCHUSTER D I, et al. Virtualized logical qubits: A 2.5D architecture for error-corrected quantum computing[C]//Proceedings of the 53rd Annual IEEE/ACM International Symposium on Microarchitecture. New York: IEEE, 2020: 173-185.
[48] ROSENBERG D, KIM D, DAS R, et al. 3D integrated superconducting qubits[J]. NPJ Quantum Information, 2017, 3: 42.
[49] MUKAI H, SAKATA K, DEVITT S J, et al. Pseudo-2D superconducting quantum computing circuit for the surface code: Proposal and preliminary tests[J]. New Journal of Physics, 2020, 22(4): 043013.
[50] AZAD U, PAPNEJA A, SAINI R, et al. Circuit centric quantum architecture design[J]. IET Quantum Communication, 2021, 2(1): 14-25.
[51] LI G S, DING Y F, XIE Y, et al. Towards efficient superconducting quantum processor architecture design[C]//Proceedings of the 25th International Conference on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2020: 1031-1045.
[52] TAN B C, CONG J. Optimality study of existing quantum computing layout synthesis tools[J]. IEEE Transactions on Computers, 2021, 70(9): 1363-1373.
[53] DE MOURA L, BJØRNER N. Z3: An efficient SMT solver[EB/OL]. [2022-05-10]. https: //link. springer. com/content/pdf/10.1007/978-3-540-78800-3_24. Pdf.
[54] BJØRNER N, PHAN A D, FLECKENSTEIN L. νZ-An optimizing SMT solver[EB/OL]. [20222-05-25]. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/nbjorner-nuz.pdf.
[55] SHAFAEI A, SAEEDI M, PEDRAM M. Qubit placement to minimize communication overhead in 2D quantum architectures[C]//Proceedings of the 19th Asia and South Pacific Design Automation Conference. New York: IEEE, 2014: 495-500.
[56] WILLE R, LYE A, DRECHSLER R. Optimal SWAP gate insertion for nearest neighbor quantum circuits[C]//Proceedings of the 19th Asia and South Pacific Design Automation Conference. New York: IEEE, 2014: 489-494.
[57] MURALI P, BAKER J M, JAVADI-ABHARI A, et al. Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers[C]//Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2019: 1015-1029.
[58] MURALI P, LINKE N M, MARTONOSI M, et al. Full-stack, real-system quantum computer studies[EB/OL]. [2023-06-25]. https://arxiv.org/pdf/1905.11349.
[59] MURALI P, LINKE N M, MARTONOSI M, et al. Full-stack, real-system quantum computer studies: Architectural comparisons and design insights[C]//Proceedings of the ACM/IEEE 46th Annual International Symposium on Computer Architecture. New York: IEEE, 2019: 527-540.
[60] BHATTACHARJEE A, BANDYOPADHYAY C, WILLE R, et al. A novel approach for nearest neighbor realization of 2D quantum circuits[C]//Proceedings of the IEEE Computer Society Annual Symposium on VLSI. New York: IEEE, 2018: 305-310.
[61] SAKI A A, ALAM M, GHOSH S. QURE: Qubit re-allocation in noisy intermediate-scale quantum computers[C]//Proceedings of the 56th ACM/IEEE Design Automation Conference. New York: IEEE, 2019: 1-6.
[62] SMITH K N, THORNTON M A. A quantum computational compiler and design tool for technology-specific targets[C]//Proceedings of the 46th International Symposium on Computer Architecture. New York: ACM, 2019: 579-588.
[63] SIRAICHI M Y, DOS SANTOS V F, COLLANGE C, et al. Qubit allocation as a combination of subgraph isomorphism and token swapping[J]. Proceedings of the ACM on Programming Languages, 2019, 3: 1-29.
[64] LI G S, DING Y F, XIE Y, et al. Tackling the qubit mapping problem for NISQ-era quantum devices[C]//Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2019: 1001-1014.
[65] DING Y S, GOKHALE P, LIN S F, et al. Systematic crosstalk mitigation for superconducting qubits via frequency-aware compilation[C]//Proceedings of the 53rd Annual IEEE/ACM International Symposium on Microarchitecture. New York: IEEE, 2020: 201-214.
[66] TANNU S S, QURESHI M K. Mitigating measurement errors in quantum computers by exploiting state-dependent bias[C]//Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture. New York: ACM, 2019: 279-290.
[67] TANNU S S, QURESHI M. Ensemble of diverse mappings: Improving reliability of quantum computers by orchestrating dissimilar mistakes[C]//Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture. New York: ACM, 2019: 253-265.
[68] HUANG Y P, MARTONOSI M. Statistical assertions for validating patterns and finding bugs in quantum programs[C]//Proceedings of the 46th International Symposium on Computer Architecture. New York: ACM,

2019: 541-553.
[69] LIU J, ZHOU H Y. Systematic approaches for precise and approximate quantum state runtime assertion[C]// Proceedings of the IEEE International Symposium on High-Performance Computer Architecture. New York: IEEE, 2021: 179-193.
[70] FOWLER A G, MARIANTONI M, MARTINIS J M, et al. Surface codes: Towards practical large-scale quantum computation[J]. Physical Review A, 2012, 86(3): 032324.
[71] DING Y S, HOLMES A, JAVADI-ABHARI A, et al. Magic-state functional units: Mapping and scheduling multi-level distillation circuits for fault-tolerant quantum architectures[C]//Proceedings of the 51st Annual IEEE/ACM International Symposium on Microarchitecture. New York: IEEE, 2018: 828-840.
[72] JAVADI-ABHARI A, GOKHALE P, HOLMES A, et al. Optimized surface code communication in superconducting quantum computers[C]//Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture. New York: IEEE, 2017: 692-705.
[73] WU B J, HE X Y, YANG S, et al. Optimization of CNOT circuits under topological constraints[EB/OL]. [2023-06-29]. https://www.xueshufan.com/publication/3195067514.
[74] CHEN J X, ZHANG F, HUANG C, et al. Classical simulation of intermediate-size quantum circuits[EB/OL]. [2023-07-5]. https://arxiv.org/pdf/1805.01450.
[75] SMELYANSKIY M, SAWAYA N P D, ASPURU-GUZIK A. qHiPSTER: The quantum high performance software testing environment[EB/OL]. [2023-07-08]. https://www.xueshufan.com/publication/2288904158.
[76] JONES T, BROWN A, BUSH I, et al. QuEST and high performance simulation of quantum computers[J]. Scientific Reports, 2019, 9: 10736.
[77] PEDNAULT E, GUNNELS J A, NANNICINI G, et al. Pareto-efficient quantum circuit simulation using tensor contraction deferral[EB/OL]. [2023-07-10]. https://www.xueshufan.com/publication/3080724058.
[78] LI R L, WU B J, YING M S, et al. Quantum supremacy circuit simulation on sunway TaihuLight[J]. IEEE Transactions on Parallel and Distributed Systems, 2020, 31(4): 805-816.
[79] ZHANG P, YUAN J B, LU X W. Quantum computer simulation on multi-GPU incorporating data locality[M]//Algorithms and Architectures for Parallel Processing. Cham: Springer International Publishing, 2015: 241-256.
[80] CHEN Y T, FARQUHAR C, PARRISH R M. Low-rank density-matrix evolution for noisy quantum circuits[J]. NPJ Quantum Information, 2021, 7: 61.
[81] LIU L, DOU X L. QuCloud+: A holistic qubit mapping scheme for single/multi-programming on 2D/3D NISQ quantum computers[J]. ACM Transactions on Architecture and Code Optimization, 2022, 21: 1-27.

编 辑 叶 芳