计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2021.20210314
58(9): 1856-1874, 2021

面向超导量子计算机的程序映射技术研究

窦星磊 刘 磊 陈岳涛
(计算机体系结构国家重点实验室(中国科学院计算技术研究所) 北京 100190)
(中国科学院计算技术研究所 北京 100190)
(liulei2010@ict.ac.cn)

An Investigation into Quantum Program Mapping on Superconducting Quantum Computers

Dou Xinglei, Liu Lei, and Chen Yuetao
(*State Key Laboratory of Computer Architecture* (*Institute of Computing Technology*, *Chinese Academy of Sciences*), *Beijing* 100190)
(*Institute of Computing Technology*, *Chinese Academy of Sciences*, *Beijing* 100190)

Abstract Errors occur due to noise when quantum programs are running on a quantum computer. Previous quantum program mapping solutions map a specific quantum program onto the most reliable region on a quantum computer for higher fidelity. Mapping multiple quantum programs onto a specific quantum computer simultaneously improves the throughput and resource utilization of the quantum computer. However, due to the scarcity of robust resources and resource allocation conflict, multi-programming on quantum computers leads to a decline in overall fidelity. We introduce quantum program mapping, classify the related studies, and analyze their characteristics and differences. Furthermore, we propose a new mapping solution for mapping concurrent quantum programs, including three key designs. 1) We propose a community detection assisted qubit partition (CDAQP) algorithm, which partitions physical qubits for concurrent quantum programs according to both physical topology and the error rates, improving the reliability of initial mapping and avoiding the waste of robust resources. 2) We introduce inter-program SWAPs, reducing the mapping overheads of concurrent quantum programs. 3) A framework for scheduling quantum program mapping tasks is proposed, which dynamically selects concurrent quantum programs to be executed, improving the throughput while ensuring the fidelity of the quantum computers. Our approach improves the fidelity by 8.6% compared with the previous solution while reducing the mapping overheads by 11.6%. Our system is a prototype of the OS for quantum computers—QuOS.

Key words quantum computing; multi-programming; mapping; fidelity; task scheduling

摘 要 量子程序在量子计算机上执行时可能由于噪声产生错误.先前的量子程序映射策略将量子程序映射至量子计算机中的最健壮的区域上,以获得更高的保真度.在量子计算机上同时映射多个量子程序可以提升量子计算机的通量和资源利用率.但由于健壮资源稀缺、资源分配冲突,并发量子程序映射会

收稿日期:2021-04-01;**修回日期**:2021-07-06
基金项目:国家自然科学基金项目(62072432,61502452)
This work was supported by the National Natural Science Foundation of China (62072432, 61502452).
通信作者:刘磊(lei.liu@zoho.com)

导致整体可靠性下降.介绍了量子程序映射,对相关研究进行分类,并深入分析了其特点与区别.此外,针对并发量子程序映射问题提出了一种新的映射策略,包括3个关键设计:1)提出了社区发现辅助量子位划分算法.结合拓扑结构和错误率数据为并发量子程序进行物理量子位划分,提升初始映射可靠性,避免健壮资源的浪费.2)引入了跨程序SWAP操作,降低了并发量子程序的映射开销.3)提出了一种量子程序映射任务的调度框架,用于动态选取并发量子程序,在保证量子计算机保真度的前提下,提升了通量.所提策略较先前工作在程序执行保真度上提升了8.6%,节省了11.6%的映射开销.所设计的系统是一个面向量子计算机的操作系统原型——QuOS.

关键词 量子计算;并发程序;映射;保真度;任务调度

中图法分类号 TP302.7

量子计算在近年来蓬勃发展,它针对特定计算任务展现出了强大的计算加速能力,可用于加速求解经典计算中的难解问题,如数据库搜索[1]、机器学习[2-3]、化学反应模拟[4-5]、加密解密[6]、优化问题[7]等.量子计算机(量子芯片)是实施量子计算的物理设备,量子位是量子计算机的基本运算单元,量子门操作用于操控量子位状态,由量子门操作序列组成的量子程序可完成特定的量子计算任务.Intel,Google,IBM等企业已相继发布了5~72量子位的量子计算机[8-10].IBM提供了量子计算云服务[11],用户可通过接口向云服务提交执行量子计算任务.在过去数10年中,量子计算的理论研究取得了较大进展;但量子计算机物理机的相关研究却始终难以克服噪声问题带来的影响.现在的量子计算机属于中等规模带噪声量子计算机(noisy intermediate-scale quantum computer,NISQ computer)类型的量子计算机[12].在该类量子计算机中,量子位状态不稳定,量子门操作可能出现错误.虽然量子纠错编码可以使量子计算机获得容错能力,但其成本太高(构造一个容错量子位需要20~100个物理量子位),无法在现阶段的量子计算机上实施.

量子计算机有超导[13]、离子阱[14]、光量子[15]等物理实现方式.本文主要关注超导量子计算机.面向超导量子计算机的量子程序映射策略可将高级语言编写的量子算法转换为能在超导量子计算机上直接执行的量子门操作指令序列.通常,量子程序映射策略的优化目标包括:1)将量子程序映射至具有低错误率的量子位和连接上执行,以提升量子程序的执行保真度.2)缩减映射过程中插入的SWAP操作数,降低映射后量子线路的复杂度,间接提升执行保真度.3)缩减映射后量子线路深度,以减少由较短的量子位状态保持时间引发的错误.

量子计算云服务的出现和兴起为传统量子程序映射策略[16-19]带来了新的挑战.传统量子程序映射策略通常仅支持映射单个量子程序.在执行量子程序时,量子计算机上未被分配的物理量子位会被闲置,这导致量子计算机中的资源未得到充分利用.此外,对量子计算云服务的访问请求越来越多,延长了服务等待时间.因此有必要通过并发机制,同时映射、执行多个量子程序来提升量子计算机的通量和资源利用率.然而并发量子程序映射带来了不可忽视的可靠性下降的问题.这是由于:量子计算机中健壮资源稀缺,不足以为每个量子程序分配足够的健壮资源;并发量子程序之间的串扰[20];映射并发量子程序时,启发式搜索的搜索空间被限制,造成更高的量子程序映射开销.先前的工作[21]表明,同时执行2个并发量子程序会导致平均保真度降低12%.

本文重点讨论面向超导量子计算机的量子程序映射技术,对先前技术进行归纳、分析和对比.针对并发量子程序映射问题,本文提出一种新的并发量子程序映射策略,在提升量子计算机通量的前提下尽可能保证量子程序的执行保真度.本文的主要贡献可以归纳为3点:

1) 深入分析了近年来面向超导量子计算机的量子程序映射工作,并进行分类,深入剖析了不同类别映射策略的特点.

2) 揭示了并发量子程序映射可能导致映射成本提升和保真度下降的问题;据此设计并发量子程序映射策略,包括社区发现辅助的量子位划分(community detection assisted qubit partition,CDAQP)算法、引入跨程序SWAP操作的映射转换算法X-SWAP,帮助提升并发量子程序的执行保真度,降低映射成本.

3) 提出了量子程序映射任务调度器,可从待执行任务队列中选择最佳量子程序组合并发执行;允许在量子计算机通量和保真度2方面进行权衡.

本文设计的系统是一个初步的面向 NISQ 量子计算机的操作系统原型(quantum operating system, QuOS).

1 量子计算概述

1.1 量子计算

量子位是量子计算机中的基本运算单元.如图 1 所示,布洛赫球面用于表示单个量子位的状态.布洛赫球面上下 2 个极点分别代表 2 个量子计算基态:$|0\rangle$和$|1\rangle$.一个量子位的状态为 2 个基态$|0\rangle$和$|1\rangle$的叠加,即$|\psi\rangle=\alpha|0\rangle+\beta|1\rangle$.其中,$\alpha$ 和β 为复数,且满足$|\alpha|^2+|\beta|^2=1$,α 和β 分别为该量子位$|0\rangle$和$|1\rangle$两个量子基态对应的概率幅.当采用读出操作对量子位状态进行测量时,测量结果为 0 的概率为$|\alpha|^2$,测量结果为 1 的概率为$|\beta|^2$.类似地,2 个量子位的状态可以表示为 4 个量子计算基态的叠加:$|\psi\rangle=\alpha_{00}|00\rangle+\alpha_{01}|01\rangle+\alpha_{10}|10\rangle+\alpha_{11}|11\rangle$.单量子位门操作(如 H,X,Y,Z 门等)可改变单个量子位的状态.受控非(controlled-not, CNOT)门为双量子位门操作,它作用于 2 个量子位.若其中控制量子位被置为 1,目标位的状态将会被翻转.涉及 3 个或更多量子位的门操作可被分解为单量子位门和双量子位门操作的组合序列[22].

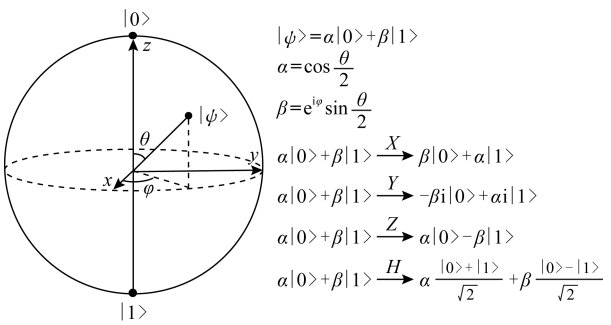


Fig. 1 The Bloch representation of a single qubit

图 1 单量子位的布洛赫球面表示

1.2 量子计算机

IBM 系列量子计算机属于超导量子计算机,具有基于约瑟夫森环结的 transmon 量子位[13]和基于微波的量子门操作[23].超导量子计算机仅在相邻物理量子位之间存在连接.涉及 2 个量子位的双量子位门操作仅能在具有相互连接的 2 个相邻物理量子位上执行.图 2 展示了 IBMQ Melbourne(后文称 IBM-Q16)超导量子计算机的量子位拓扑结构.

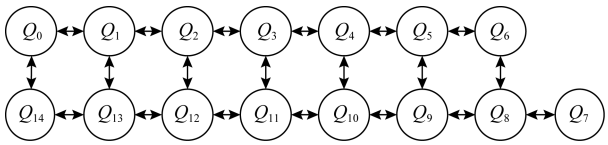


Fig. 2 Qubit topology of IBMQ Melbourne

图 2 IBMQ Melbourne 的量子位拓扑结构

1.3 量子计算机中的错误

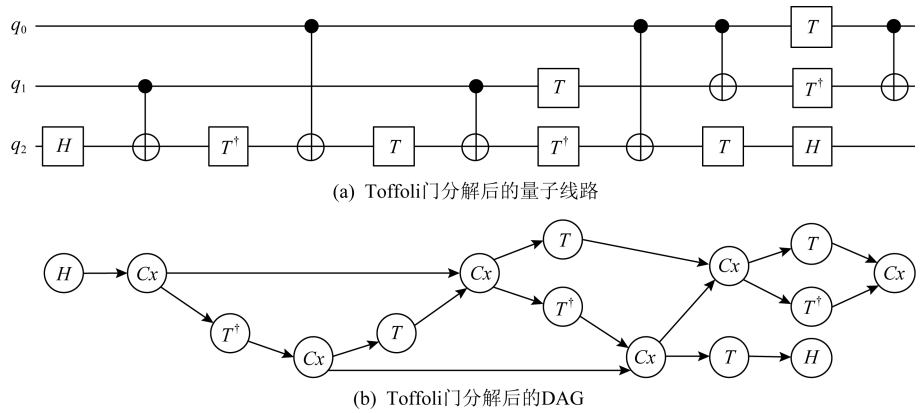
由于量子计算机的物理量子位状态不稳定,量子位间的连接脆弱且易受干扰.在量子计算机上运行的量子程序可能会发生 3 种类型的错误:1)由较短的量子位状态保持时间导致的保持错误.保持错误主要受到在量子计算机上执行的量子程序深度的影响.量子程序深度越深,执行时间越长,可靠性就越低.2)由易错的量子门操作导致的操作错误.操作错误主要受到在量子计算机上执行的量子程序复杂度的影响.量子程序中包含的门操作越多,出错的概率越高.3)测量操作带来的读出错误.量子程序在执行结束后需要执行测量操作以获取量子位状态.测量操作不可靠,可能读出错误的状态.

IBM 量子计算云服务提供了错误率标定数据,可通过接口获取.标定数据约每 24 h 更新一次.

1.4 量子程序映射

量子程序由一系列量子门操作组成,可用量子线路表示.图 3(a)展示了三量子位 Toffoli 门操作分解为单、双量子位门操作序列后的量子线路.其中每条横线代表一个逻辑量子位,横线上的节点代表单、双量子位门操作.图 3(b)展示了该量子线路的有向无环图(directed acyclic graph, DAG).DAG 用于表示量子程序中量子门操作的执行顺序依赖关系,其中每个节点代表一个量子门操作.当一个量子门操作在 DAG 中有前驱节点未被执行时,该量子门操作无法执行.当一个量子门操作在 DAG 中没有未执行的前驱节点时,它在逻辑上才可被执行.

求解量子程序映射问题需要遵循 3 个约束:1)每个逻辑量子位都需要被映射至量子计算机中的一个物理量子位上,且该物理量子位不能再映射其他的逻辑量子位.2)若一个量子门操作未执行,其后继的量子门操作也不能被执行.3)当一个双量子位门操作涉及到的逻辑量子位的映射位置在物理上相邻时,该双量子位门操作才能被执行,否则可在量子线路中插入 SWAP 操作(一个 SWAP 操作由 3 个 CNOT 操作组成,用于交换 2 个逻辑量子位的映射位置),使涉及到的逻辑量子位在物理上相邻.



(a) Toffoli门分解后的量子线路

(b) Toffoli门分解后的DAG

Fig. 3 The quantum circuit and DAG of decomposed Toffoli gate

图 3 Toffoli 门分解后的量子线路和 DAG

解决量子程序映射问题一般包括构建初始映射和映射转换 2 个步骤:

1) 构建初始映射.即把每个逻辑量子位映射至一个物理量子位.初始映射对量子程序的后续映射成本和执行保真度具有重要影响.初始映射需要满足:程序被分配在量子计算机中最健壮的量子位上;初始分配的量子位邻近,便于节约后续映射成本.

2) 映射转换.即在量子线路中插入 SWAP 操作,满足所有双量子位门操作的逻辑量子位映射位置约束,使所有双量子位门操作均可执行.映射转换时插入的 SWAP 操作会导致量子程序执行保真度降低,因此需保证插入的 SWAP 操作尽可能少,且尽可能使用量子计算机中的健壮资源.

1.5 量子操作系统

量子操作系统是量子计算机中软硬件资源的管理者,负责管理、配置具有不同物理实现方式的量子位;调度量子程序;为量子程序提供最可靠的映射方案.量子计算机操作系统与经典计算机操作系统在设计原则和设计目标上具有较大区别.本文设计的系统是一个初步的面向 NISQ 量子计算机的操作系统原型 QuOS.在后续工作中,作者将进一步对量子操作系统和量子计算技术栈中的其他技术进行探索.

2 量子程序映射技术概览

量子程序映射技术属于量子计算技术栈[24]中的软件栈层次,位于高级语言(用于表示特定的量子算法)和量子计算机系统结构(包括量子位、量子门操作、量子纠错等)层次之间.量子计算软件栈覆盖范围广,涉及多量子位门操作分解[22,25]、量子线路映射前/后优化[26-27]、量子程序调试与断言[28-29]、量子缓存[30]、并发量子程序映射[21,31-32]等问题.本文针对量子程序映射问题进行研究,假设量子线路中的多量子位门操作已被分解,线路中的门操作均能在目标量子计算机上执行;不进行映射前/后优化;仅考虑在量子线路中插入 SWAP 操作,满足所有双量子位门操作的逻辑量子位映射位置约束.

量子程序映射问题已被证明为 NP-完全问题(NP-Complete problem)[33-34].文献[35]综述了常见的量子计算机物理实现技术及相应的量子位拓扑结构,包括一维、二维拓扑结构.针对具体的物理量子位拓扑结构,以保真度或插入 SWAP 数为优化目标的量子程序映射策略陆续被提出.这些量子程序映射技术可分为 2 类:

1) 基于数学问题求解器的最优化方法.该类方法可根据量子程序映射问题设计约束条件和优化目标,将量子程序映射问题转换为等价的数学优化问题,并采用求解器求解.这类方法可求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,运行时间长,无法支持求解大型量子程序(具有数十个以上量子位的量子程序)映射问题.此外,该类方法难以同时对多个优化目标进行优化,且难以实现多个优化目标之间的权衡.

2) 启发式方法.启发式方法采用启发式信息辅助搜索.量子程序映射策略依据启发式信息或贪心策略确定逻辑量子位初始映射位置,后根据启发式信息筛选出最合适的 SWAP 操作插入程序进行映射转换.启发式方法更为灵活,具有更佳的可扩展性,不受限于量子芯片和量子程序的规模,有处理更大规模量子程序映射问题的能力.然而启发式策略无法保证求解所得的结果为最优解.可能存在保真度更高、映射成本更低的映射方案.

2.1 最优化方法

本文对具有代表性的最优化方法进行了整理汇总,如表1所示.现有工作中,量子程序映射问题可等价转化为如下类型数学问题:可满足性(Boolean satisfiability, SAT)问题、可满足性模理论(satisfiability modulo theories, SMT)问题、伪布尔优化(pseudo Boolean optimization, PBO)问题、动态规划(dynamic programming, DP)问题、时序规划(temporal planning, TP)问题、约束规划(constrained planning, CP)问题、整数线性规划(integer linear programming, ILP)问题、混合整数规划(mixed integer programming, MIP)问题等.

Table 1 Optimization Methods for Quantum Program Mapping

表1 量子程序映射最优化方法

文献方法	时间	转换问题	优化目标	特点
Shafaei 等人的工作[36]	2014	MIP	SWAP 数	针对 2D 架构优化,采用启发式方法优化求解器求解结果
Wille 等人的工作[37]	2014	PBO/SAT	SWAP 数	构建了量子程序映射问题到 PBO 问题的转换,为启发式与最优化方法的对比提供依据
Lye 等人的工作[38]	2015	PBO	SWAP 数	对比了启发式方法和最优化方法实例
Booth 等人的工作[39]	2018	TP/CP	线路深度/SWAP 数	结合 2 种求解器的优点,提供高质量的映射方案
Qubit Allocation[33]	2018	DP	SWAP 数	采用动态规划计算,存储计算中间结果,避免重复运算
MUQUT[40]	2019	ILP	线路深度/SWAP 数	结合求解器与启发式方法,提升执行保真度
T/R-SMT[18]	2019	SMT	线路深度/保真度	将错误率引入优化目标,提升执行保真度
IBM QX Mapping for Minimal Mapping[41]	2019	SMT	SWAP 数	缩小搜索空间,提升执行效率,保证近似最优解
TriQ[42]	2019	SMT	保真度	在多平台上对比实验,揭示了量子计算机结构对性能的影响
XtalkSched[20]	2020	SMT	保真度	将串扰错误率加入优化目标,缓解串扰错误
(TB-)OLSQ[43]	2020	SMT	线路深度/SWAP 数/保真度	采用近似策略提升求解效率;支持多种优化目标

若从量子程序映射问题转化为等价数学优化问题,需转化的内容一般包括:1)待优化变量.量子程序映射过程中,逻辑量子位的映射位置、门操作的执行时刻、SWAP 路径均需转换为等价数学优化问题中的待优化变量,以便于求解.2)约束条件.须将量子程序映射问题中的限制条件(如门操作需顺序执行、每个物理量子位仅允许映射一个逻辑量子位等)转换为等价数学优化问题中的限制条件.3)优化目标.优化目标通常包括程序执行保真度、映射过程中插入的 SWAP 操作数目、映射后线路深度(或程序执行时间)3 种.

文献[36]将量子程序映射问题转化为 MIP 问题.该策略将量子线路按照数据依赖关系划分为多个层,每层中的门操作不涉及相同的逻辑量子位,可同时执行.该映射策略首先确定初始映射,然后利用求解器确定相邻 2 层之间的映射转换 SWAP 操作.映射所需的 SWAP 数目作为优化目标.MUQUT[40]根据量子程序映射问题构造 ILP 问题,并采用求解器 Gurobi[44]求解.该策略同样采用将量子线路分层并在层间插入 SWAP 的方法进行量子程序映射.线路深度作为优化目标.该策略将量子程序映射位置所占矩形区域在量子芯片上滑动,来搜索可靠性最高的映射位置.

Wille 等人[37]实现了量子程序映射问题到 PBO 问题的转化,实现了最优化方法与启发式方法的对比,为启发式方法结果的评价提供了依据,并在后续工作中[38]给出了与启发式方法对比的实际案例.

Booth 等人[39]将量子程序映射问题转换为 TP 问题,结合线路深度和插入 SWAP 数,构建加权优化目标函数.该策略将 TP 问题和 CP 问题进行结合.首先求解 TP 问题,提供可靠的初始解决方案,在此基础上求解 CP 问题,进一步提升解决方案的质量.

Siraichi 等人[33]对比了最优化方法与启发式方法,并给出了基于 DP 的最优化量子程序映射算法实现.DP 执行过程中可存储中间结果,避免重复计算,提升计算效率.

文献[18,20,41-43]采用 SMT 求解器求解量子程序映射问题.常见 SMT 求解器包括 Z3[45],vZ[46].文献[18]分别设计了 T-SMT 和 R-SMT.T-SMT 以程序执行时间为优化目标,使门操作序列中最后一个门操作的完成时间最小化.而 R-SMT 以程序执行保真度为优化目标,结合 CNOT 门和读出操作错误率数据,设计了 CNOT 错误率与读出操作错误率的加权目标函数,允许为不同种类的错误分配不同的

权重.文献[20]对量子程序执行时的串扰错误进行了精准建模,并将串扰错误率结合至目标函数中,采用 SMT 求解器最小化串扰错误.文献[43]介绍了一种最优化方法 OLSQ 和近似最优化方法 TB-OLSQ,可对程序执行保真度、插入 SWAP 数目、映射后线路深度三者中任一优化目标进行优化.TB-OLSQ 在可被连续执行的门操作集合之间插入 SWAP 操作,提升求解器求解效率,求得近似最优解,保证程序执行的保真度.TB-OLSQ 针对量子近似优化算法(quantum approximate optimization algorithm, QAOA)进行了特殊优化,可缩减映射成本及映射后线路深度.TriQ[42]构建了两两物理量子位之间交互的可靠性矩阵,并据此转化量子程序映射问题为 SMT 问题,优化程序的执行保真度.TriQ 对比了多个量子程序在不同量子计算平台上的性能,揭示了量子芯片结构对性能的影响.

2.2 启发式方法

启发式方法的代表性工作如表 2 所示.针对目标平台,启发式方法可分为 2 类:

1) 针对简化模型的启发式算法.该类算法通常将量子芯片的拓扑结构简化为 1D 线性模型或 2D 网格模型.Shafaei 等人依据最小线性排序(minimum linear arrangement, MinLA)问题,设计启发式策略[47],缩减映射时所需的 SWAP 操作数目.该策略将量子线路分割为多个子图,并在每个子图中应用 MinLA 问题求解最佳 SWAP 路径.Wille 等人[48]采用前瞻算法,对 1D 和 2D 两种简化模型,优化插入的 SWAP 操作数,降低了 56% 的 SWAP 成本.Bhattacharjee 等人[49]面向 2D 量子芯片网格模型,以插入的 SWAP 操作数为优化目标,设计新的启发式量子程序映射策略.通过 3 个步骤:选择量子位、映射量子位和插入 SWAP,降低了映射开销.QURE[17]在为量子程序确定初始映射后,搜索 2D 网格模型中的同构子图.即在量子芯片上滑动包括了映射位置的矩形区域,预估不同位置下量子程序的执行保真度.选择可靠性最高的区域作为量子程序的最终映射区域.

2) 针对特定量子芯片结构的启发式算法.该类算法可根据具体量子芯片架构和错误率数据完成映射.Siraichi 等人提出启发式方法[33],相比于最优化方法,能处理更大规模的量子程序映射问题.该策略在初始映射中,最大化已满足映射相邻关系的双量子位门数,之后插入 SWAP 进行映射转换.Zulehner 等人针对任意量子线路,设计启发式映射策略[50].类似于文献[36,40],该策略首先根据数据依赖关系将量子线路划分为多个相互独立的层,每个层中的门操作可同时执行;之后采用 A* 算法搜索相邻层之间成本最低的映射转换 SWAP 操作.此后,Zulehner 等人针对特殊的 SU(4)程序对映射策略进行了改进[27].该策略通过预处理步骤,对程序内门操作进行分组,减小映射复杂度.之后采用 A* 算法进行映射,并进行映射后优化,提升 SU(4)程序执行保真度.Tannu 等人的工作[19]在文献[50]的基础上,引入了对量子计算机中错误率数据的分析,提出了考虑噪声的量子程序映射算法,利用了健壮资源,提升了程序执行保真度.Smith 等人[51]利用连通树映射

Table 2 Heuristic Methods for Quantum Program Mapping

表 2 量子程序映射启发式方法

文献方法	时间	目标平台	特点
Shafaei 等人的工作[47]	2013	简化模型	针对一维芯片模型,将量子程序映射问题转化为图排序问题求解
Wille 等人的工作[48]	2016	简化模型	采用前瞻算法,优化 1D/2D 网格结构的 SWAP 成本
Bhattacharjee 等人的工作[49]	2018	简化模型	针对 2D 网格结构,采用启发式策略缩减了映射成本
QURE[17]	2019	简化模型	采用同构子图搜索方法最大化保真度
Qubit Allocation[33]	2018	特定芯片	初始映射最大化已满足映射相邻关系的 CNOT 操作数
IBM QX Mapping for Arbitrary Circuits[50]	2018	特定芯片	采用 A* 算法,寻找低成本的层间映射转换
IBM QX Mapping for SU(4) Circuits[27]	2019	特定芯片	针对 SU(4)量子线路进行映射策略优化
VQA/VQM[19]	2019	特定芯片	考虑错误率数据,将量子程序映射至健壮资源上执行
Smith 等人的工作[51]	2019	特定芯片	采用连通树映射算法寻找 SWAP 路径,在多个平台上进行评测
Siraichi 等人的工作[52]	2019	特定芯片	基于子图同构和令牌交换建模量子位映射问题,简化映射问题
SABRE[16]	2019	特定芯片	设计更高效的启发式搜索空间,提供指数级别加速
Greedy V/E[18]	2019	特定芯片	采用不同策略进行初始映射,提升执行保真度
SQUARE[53]	2020	特定芯片	采用启发式方法动态分配、回收量子位,优化资源占用

算法,寻找最优 SWAP 路径,并在多个量子计算平台上进行了评测,策略适用于不同的量子芯片拓扑结构.Siraichi 等人[52]将量子位映射问题转化为子图同构问题和令牌交换问题的组合,简化量子位映射问题,利用已有的启发式策略进行求解.SABRE[16]通过对量子程序映射转换问题的精准建模,缩减了启发式搜索空间,提供了更有效的 SWAP 操作,为量子程序映射算法带来了指数级别的加速.Murali 等人的工作[18]中进行了不同种启发式策略和最优化策略的对比.对比了 2 种启发式映射方案的性能,分别为:优先映射最频繁使用的量子位和优先映射最频繁使用的连接.该工作认为启发式方法具有更高的可扩展性,具有处理更大型量子程序映射问题的能力.Ding 等人提出启发式算法[53],指导动态分配和回收量子位,利用量子位的局部性、量子程序的模块性和并行性,复用量子位,将量子程序的执行保真度提升了 1.47 倍.

3 基于社区划分的初始映射构建

3.1 先前策略的问题

量子程序的初始映射极为关键.对于单个量子程序而言,合适的初始映射可以帮助节约后续映射成本,提升量子程序的执行保真度.而对于并发量子程序映射问题,合适的初始映射不仅可以缩减映射成本,还可以尽可能地利用量子计算机中的健壮资源,避免资源浪费,减小并发量子程序之间的干扰,提升整体执行保真度.

量子计算机中健壮资源分布和物理拓扑结构各不相同,先前的量子位初始映射策略通常利用贪心算法或启发式搜索算法,将一个量子程序分配至量子计算机中最健壮的区域.图 4 中虚线连接代表具有高错误率的不可靠连接,虚线框代表量子程序的映射位置.若一个具有 4 个量子位的量子程序需要被映射至如图 4 所示的量子计算机上,它将会被映射至物理量子位 Q_1,Q_2,Q_5,Q_6.映射策略通常选择具有可靠连接的物理量子位,以提升执行保真度.

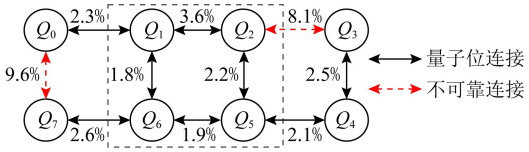


Fig. 4 An example for qubit mapping

图 4 量子位映射示例

为了提升量子计算机的通量和资源利用率,并发量子程序映射策略应运而生[21].然而并发量子程序映射也为传统量子程序映射策略带来了新的挑战.尽管可以将并发量子程序合并为一个量子程序,采用传统的面向单个量子程序的映射策略[16,18]进行映射.但传统映射策略并不适合处理并发量子程序映射任务,可能导致 3 个问题:1)无法保证并发量子程序的执行保真度.量子计算机中健壮资源有限,无法保证不同规模的量子程序都被分配足够健壮的资源.2)并发量子程序数目无法在线进行调整.例如当并发量子程序执行时,其中一个量子程序保真度大幅降低,传统映射策略无法将并发量子程序组合回退为单独执行.3)传统映射策略无法利用并发量子程序映射问题中新的优化契机.例如,采用跨程序 SWAP 操作可减少并发量子程序的映射成本.

现有的并发量子程序映射策略[21]允许并发执行 2 个量子程序,但其资源利用率和执行保真度仍有待提升.下面在图 5 中以一个例子来说明现有的并发量子程序映射策略面临资源利用率低的问题.使用现有的并发量子程序映射算法时,大规模的健壮物理量子位集合可能会被割裂为规模较小的量子位集合碎片.这些碎片中仅包含很少的物理量子位,不足以映射任何量子程序,导致健壮资源未被利用.

图 5 展示了采用先前工作[21]中提出的 FRP 策略构建量子程序初始映射的例子.图 5 中有 2 个量子程序 P_1(5 量子位)和 P_2(4 量子位)需被映射至 IBM-Q16 上.图 5 中展示了量子计算机 IBM-Q16 的

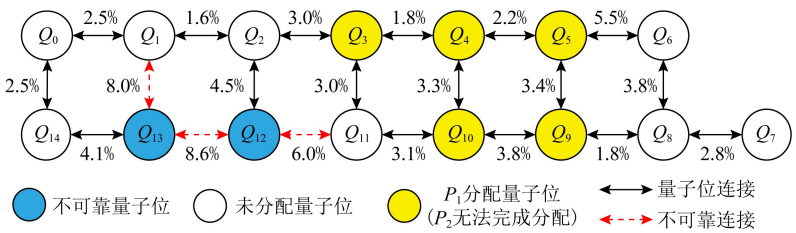


Fig. 5 Previous initial mapping technique incurs resource under-utilization

图 5 先前的初始映射策略导致资源利用率低

物理结构图,其错误率标定数据采集于 2019 年 12 月 27 日.具有较高错误率的不可靠连接在图中被标注为虚线,不可靠的量子位同样进行了标注.P_1 先被映射.FRP 策略从量子计算机中的一个具有足够多高 *utility* 值的邻居的量子位开始,扩展分配区域.该策略定义指标 *utility* 为:一个量子位所包含的连接数/连接上 CNOT 操作错误率之和[21].因此,物理量子位 Q_3,Q_4,Q_5,Q_9 和 Q_{10} 被用于映射量子程序 P_1.P_1 映射完成后,映射策略无法为 P_2 找到映射位置.因为在 $Q_0 \sim Q_2$,$Q_{11} \sim Q_{14}$ 中无法找到一个具有足够多健壮邻居的分配起点;而 $Q_6 \sim Q_8$ 仅包含 3 个量子位,无法供 P_2 分配,造成了资源浪费.实际上存在更好的映射方法,能使量子计算机中的可靠资源得到充分利用,减少资源浪费.

3.2 算法设计

对于并发量子程序映射,我们观察到 3 点现象:1)量子芯片上的健壮量子位和连接是有限的.2)量子芯片上的某些量子位与周围的量子位有更多的连接.如图 2 所示,Q_1 与 3 个相邻的物理量子位相连,而 Q_7 仅与 1 个量子位相连.3)一个量子程序的量子位应当被映射到邻近的物理量子位上.并发量子程序之间的初始映射应避免相互干扰,公平利用量子计算机中的健壮资源.

本文针对并发量子程序映射问题提出了一种新的映射策略,即社区发现辅助的量子位划分(CDAQP)算法.该算法首先构造一棵由量子位组成的层次树,协助搜索量子计算机中紧密连接的健壮量子位集合.图 6 展示了 CDAQP 算法的框架.CDAQP 根据从 IBMQ 接口[11]获得的量子芯片拓扑结构和错误率标定数据构建层次树.在层次树中,叶节点表示特定的物理量子位;中间节点表示其子节点的并集.层次树构建完成之后,CDAQP 自底向上迭代层次树,查找可用于初始分配的社区.最后,根据贪心策略分配量子位.详细说明如下.

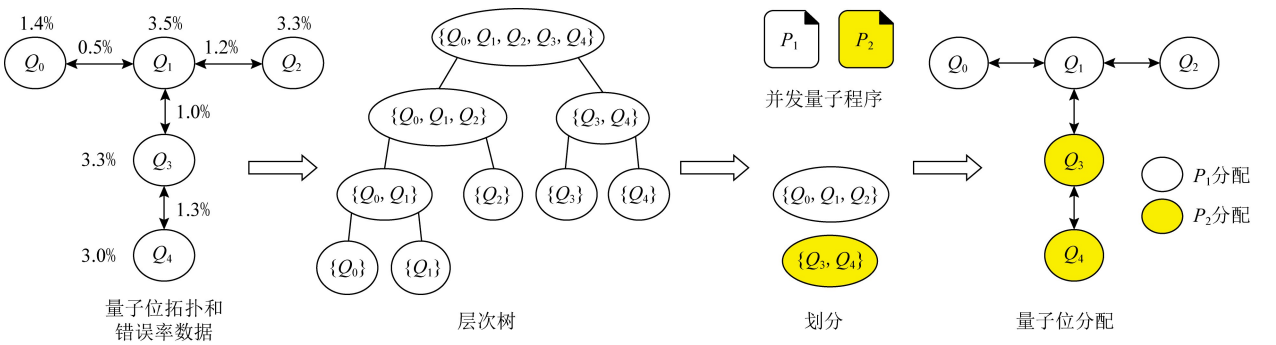


Fig. 6 Qubit mapping framework by using CDAQP algorithm

图 6 CDAQP 算法量子位映射框架

1) 层次树构建

层次树是对量子芯片中健壮资源分布的建模,有助于定位量子芯片上健壮的量子位和连接.算法 1 基于 fast-Newman(FN)社区检测算法[54]构建层次树.该算法将物理量子位划分为社区.一个社区中的物理量子位具有可靠且紧密的相互连接.相反,社区与社区之间的连接具有相对较低的可靠度.

算法 1. 层次树构建算法.

输入:量子计算机拓扑结构图、错误率标定数据;

输出:层次树 HT.

① $HT \leftarrow list()$;/* 初始化 HT 为空列表 */

② 为每个量子位在 HT 中创建一个叶节点;

③ while $HT.length > 1$ do

④ 选择 HT 中能使 $f(A, B)$值最大的 2 项 A 和 B;

⑤ 合并 A,B 为新节点 New_Node,并将 A,B 分别设置为 New_Node 的左右子树;

⑥ $HT.append(New\_Node)$;

⑦ $HT.remove(A)$;

⑧ $HT.remove(B)$;

⑨ end while

⑩ return HT.

算法初始化时,每个物理量子位对应一个社区,即层次树中的一个叶节点.算法不断合并能够使奖励函数 f 最大化的 2 个社区,直到最终只剩下一个包含所有物理量子位的社区.在该过程中,每个社区对应层次树中的一个节点,该节点是可用于分配量子位的一个候选区域.奖励函数 f 被定义为合并 2 个社区所带来的收益:

$$f = Q_{\text{merged}} - Q_{\text{origin}} + \omega EV, \tag{1}$$

其中,Q 代表一种社区划分方案的模块度(即 $Q=\sum_i (e_{ii}-a_i^2)$[54],其中 e_{ii} 代表该划方案中社区 i 内的边占量子计算机中所有边的比例,a_i 则代表所有涉及到社区 i 内节点的边占量子计算机中所有边的比例),Q 的值越高,代表着一个社区划分方案越合适.Q_{origin} 和 Q_{merged} 分别代表原划分情况和合并了 2 个社区之后的划分情况的模块度.E 代表待合并的 2 个社区间的边上 CNOT 操作的平均可靠度(可靠度为 1-操作错误率),V 代表 2 个待合并社区的量子位上的读出操作平均可靠度.奖励函数 f 使得 CDAQP 在划分时能够同时考虑物理拓扑和错误率.ω 是一个权重参数.对于特定的量子芯片,我们可以通过改变 ω 的值来调整物理拓扑与错误率间的权重.如果 $\omega=0$,则 CDAQP 完全按照物理拓扑进行划分;随着 ω 的增加,社区划分算法开始逐渐依赖于错误率数据;如果 ω 持续增加,错误率的权重将超过物理拓扑的权重,导致其退化为基于错误率的贪心算法.ω 值如何设置将在 3.3 节中进行讨论.

层次树有 3 个特点:1)层次树中的每一个节点都是可用于初始映射的候选量子位集合.2)在一个社区中的物理量子位相互连接紧密,能够帮助减小量子程序映射成本.3)具有低读出操作错误率和健壮连接的量子位会优先被合并.因此量子位集合越健壮,其在层次树中的深度就越深.层次树的这些特点能够帮助定位量子计算机上的健壮资源,为量子程序提供更优秀的初始映射.

接下来,采用图 6 中的例子解释为何层次树能帮助选择健壮的初始映射.图 6 中,左侧展示了 IBMQ London 的物理结构图,每个节点对应的数值代表该物理量子位上读出操作的错误率,在连接上的数值代表该连接上的 CNOT 操作错误率.接着展示了根据拓扑结构和错误率数据构建出的层次树.层次树构建过程为:①Q_0 和 Q_1 之间具有最低的错误率,因此它们被首先合并为一个社区.②虽然 Q_1,Q_3 间的 CNOT 操作错误率低于 Q_1,Q_2 间的 CNOT 操作错误率,但 Q_2 被首先加入社区$\{Q_0,Q_1\}$.这是因为算法倾向于将连接紧密的量子位合并为一个社区,从而避免量子计算机中健壮资源的浪费.同样,Q_3,Q_4 也被合并为一个社区.③所有的物理量子位被合并为一个社区,形成层次树的根节点.算法不仅考虑量子计算机中的资源可靠性,也同样考虑量子计算机的物理拓扑结构,选取连接紧密的量子位.这可以避免量子计算机中健壮资源的浪费,提高资源利用率,能使量子计算机映射更多的量子程序.

错误率标定数据并非随时改变,IBM 量子计算云服务约每 24 h 标定一次错误率数据.因此在错误率标定的一个周期内,构建的层次树可被存储,以供重复使用,不会造成不可接受的运算开销.

2) 量子位划分和分配

算法 2 根据层次树将量子位划分为多个区域供并发量子程序进行初始映射.算法优先为具有高 *CNOT density* 的量子程序划分区域.*CNOT density* 被定义为一个程序内 CNOT 操作数/该程序内量子位数.对于每个量子程序,算法自底向上搜索层次树,找到可供分配的量子位集合.选择具有最低平均错误率的候选量子位集合用于映射该程序.最后,使用最大权重边优先(greatest weighted edge first)策略[18]进行量子位映射.该策略将量子程序中交互数目最高的 2 个逻辑量子位映射到量子芯片上最健壮的边上,有助于生成具有高可靠性和低映射开销的初始映射.

算法 2. 量子位划分算法.

输入:层次树 *HT*、量子程序 *programs*;

输出:量子位划分 *partition*.

① 将 *programs* 按 *CNOT density* 降序排序;
② for *program* in *programs*
③ $C\leftarrow set()$;
/* 初始化候选节点 C 为空集 */
④ for l in $HT.leaves$
/* 对于 HT 中的每个叶节点 l */
⑤ $candidate\leftarrow search(l)$;/* 从 l 起自底向上搜索层次树,得到量子位数目满足待映射程序要求的节点 *candidate* */
⑥ $C.add(candidate)$;
⑦ end for
⑧ if C 为空
⑨ 失败,回退至单独执行;
⑩ end if
⑪ $node\leftarrow node\_with\_the\_max\_fidelity(C)$;
⑫ $partition.add(node)$;
⑬ 将 *node* 中的量子位从 HT 的所有节点中移除;
⑭ if *node* 的兄弟节点没有通向层次树中其他节点的路径

⑮ 将兄弟节点中包含的量子位从其祖先节点中移除;

⑯ 移除兄弟节点到其祖先节点的链接;

⑰ end if

⑱ end for

⑲ return *partition*.

3.3 权重参数选择

在式(1)中,E 表示量子位之间连接的可靠性,V 表示量子位读出操作的可靠性.该设计能帮助在每次迭代中,将具有健壮连接和较低读出错误率的可靠量子位合并到一个特定的社区中.不可靠的量子位最终才将被添加到社区中.在执行量子位分配时,CDAQP 自底向上搜索层次树,查找初始分配的候选区域.不可靠的量子位不易被选择,提高了整体可靠性.

使用 CDAQP 可能会导致分配给一个量子程序的物理量子位数目大于该量子程序真正所需的物理量子位数目.例如一个具有 4 个逻辑量子位的量子程序被映射到图 6 中所示的量子芯片上.唯一可能的分配区域是该层次树的根节点:$\{Q_0,Q_1,Q_2,Q_3,Q_4\}$.这会导致一个量子位冗余.为了避免浪费,未被使用的冗余量子位会被标记,它们可以被添加到相邻的社区,用于其他量子程序的初始映射.

对于层次树中的一个节点,本文定义最大冗余量子位数,代表当一个量子位集合(一个层次树节点)被分配给一个量子程序进行初始映射时,可能导致的最大未被使用的物理量子位数目.一个层次树节点 *node* 对应的最大冗余量子位数可计算为:$node.n\_qubits-(1+\max(node.left.n\_qubits,node.right.n\_qubits))$.

奖励函数 f 中的权重参数 ω 的增大会导致层次树的退化,即层次树的每次合并过程中,仅有一个包含一个物理量子位的叶节点被合并入新社区.这时,新社区对应的层次树节点的最大冗余量子位数为 0.即 ω 的增大可以导致平均最大冗余量子位数的减小.我们连续 21 天采集了 IBM-Q16 的错误率标定数据.对于每天的错误率标定数据,按 0.05 的步长从 0～2.5 变化 ω 的值,执行算法 1,并记录层次树中冗余量子位的平均数量,如图 7 所示.图 7 中节点颜色越深代表越多结果重叠在该位置上.采取肘部原则,取 $\omega=0.95$.在这种情况下,CDAQP 算法既考虑了物理拓扑结构,又考虑了错误率数据.平均冗余量子位数在这种情况下为 0.29.在 IBM-Q50 上进行了相同的实验,$\omega=0.40$.此外,CDAQP 不仅可以为并发量子程序提供可靠的初始映射,同样可以处理只有一个量子程序需要映射的问题.

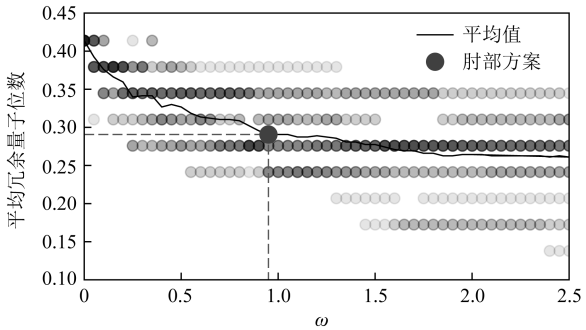


Fig. 7 Select the knee solution as ω

图 7 选取肘部方案为 ω 的值

4 跨程序 SWAP 映射转换

4.1 跨程序 SWAP 操作优势分析

并发量子程序映射问题为传统映射转换策略带来了新的挑战.当多个量子程序同时运行时,映射转换所需插入的 SWAP 数目可能增加.先前的并发量子程序映射转换策略[21]顺序处理每一个量子程序,后被处理的程序只能使用未被其他程序占用的量子位.对于一个特定量子程序来说,要使用 SWAP 操作移动量子位,使 2 个量子位相邻,最短 SWAP 路径可能涉及到多个量子位.SWAP 操作只在相邻的量子位之间进行.如果最短 SWAP 路径上的任何一个量子位被其他程序占用,将引入更多的 SWAP 操作,导致整体的可靠性下降.

优秀的映射转换策略应当能够更好地利用所有可能的 SWAP 操作,在最大程度上降低量子程序映射转换的开销.本文提出了引入跨程序 SWAP 操作的映射转换算法 X-SWAP.与先前映射转换策略不同,该算法并非单独处理每个量子程序,而是对所有并发量子程序同时进行映射转换.X-SWAP 既允许程序内 SWAP 操作,又允许跨程序 SWAP 操作.SWAP 操作涉及到的 2 个物理量子位可能分别属于 2 个量子程序(跨程序 SWAP),也可能只属于 1 个量子程序(程序内 SWAP).SWAP 所涉及到的逻辑量子位上的后续门操作都需在 SWAP 操作之后执行.

该算法允许在所有物理量子位上执行 SWAP,避免了先前策略中由于量子位被占用而导致的额外 SWAP 开销.另外,当 2 个量子程序映射相邻时,启用跨程序 SWAP 操作能够帮助缩减映射转换成本.

X-SWAP 相比于之前的映射转换策略的主要优点在于扩大了启发式搜索空间,涵盖了更多的量子位,引入更多可能的 SWAP 操作.

在下面 2 种情况下,跨程序 SWAP 操作优于程序内 SWAP 操作:

1) 1 个跨程序 SWAP 操作可以替代 2 个程序内 SWAP 操作.图 8 展示了将 2 个量子程序映射到具有 6 个物理量子位的量子计算机上的例子.图 8(a)展示了需要映射的 2 个量子程序,图 8(b)展示了 2 个量子程序的初始映射,以及在不允许跨程序 SWAP 操作情况下的映射过程.对于量子程序 P_1,g_1 和 g_2 可以直接执行,无需插入 SWAP 操作,但 g_3 却不能直接执行,需在 q_2 和 q_3 之间插入 SWAP 操作.同理,对于量子程序 P_2,为了使 g_6 能够执行,需要在 q_4 和 q_6 之间插入 SWAP 操作.

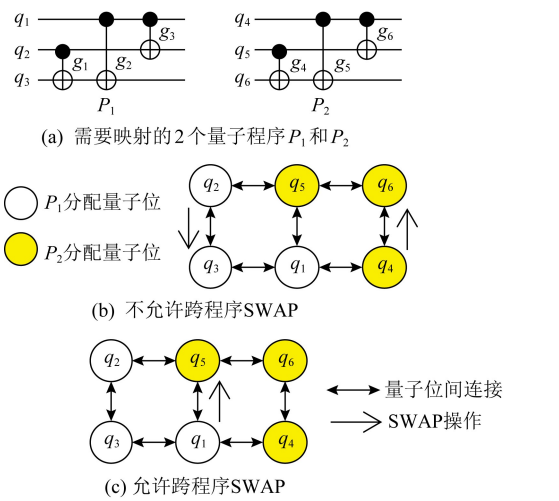


Fig. 8 An inter-program SWAP can replace two intra-program SWAPs

图 8 跨程序 SWAP 操作可替换 2 个程序内 SWAP 操作

然而,如果 2 个程序同时映射,可以引入跨程序 SWAP 操作降低量子程序的映射成本.图 8(c)展示了使用跨程序 SWAP 操作进行映射转换的例子,只需要在 q_1 与 q_5 之间插入一个 SWAP 操作,即可完成映射转换.在这种情况下,q_1 与 q_5 之间的一个 SWAP 操作代替了 2 个 SWAP 操作,降低了量子程序的映射成本,间接提升了量子程序执行保真度.

2) 在进行映射转换时,跨程序 SWAP 操作可采取捷径.图 9 中,P_1 和 P_2 被映射到具有 9 个物理量子位的量子计算机上.接下来需要处理门操作 CNOT q_1 q_5.如图 9(b)所示,先前工作中的映射转换策略[21]需引入$\{q_1,q_2\}$,$\{q_1,q_3\}$,$\{q_1,q_4\}$这 3 个 SWAP 操作才能完成映射转换,而跨程序 SWAP 操作仅需引入一个 SWAP 操作$\{q_1,q_9\}$.

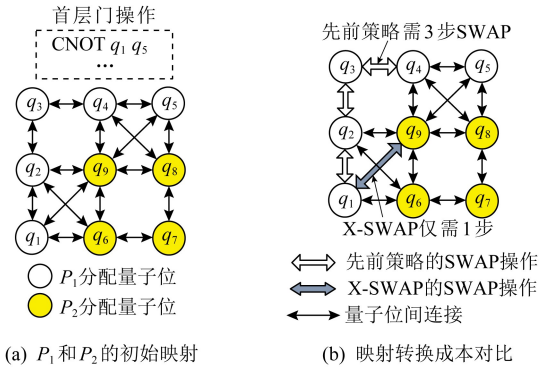


Fig. 9 Inter-program SWAPs take shortcuts

图 9 跨程序 SWAP 可采取捷径

4.2 启发式搜索空间设计

由于单量子位门操作可以直接执行,无需 SWAP 操作.因此在映射转换过程中仅考虑 CNOT 门操作.对于一个量子程序 P,图 10 展示了 P 中的关键门(critical gates, CG)操作.该量子线路的 CNOT 门操作可以分为 4 层(即 $l_1 \sim l_4$).同一层中的门操作不涉及相同的逻辑量子位,可以同时执行.l_1 是 P 的首层门操作(front layer,记为 F),它表示 P 的有向无环图(DAG)中没有未执行前驱节点的 CNOT 门操作的集合.DAG 展示了 P 的数据依赖关系.关键门操作 CG 表示 F 中在第 2 层(l_2)上拥有后继节点的门操作的集合.例如,在 F 中,g_1 在 l_2 中拥有后继节点 g_3,而 g_2 没有后继.因此,g_1 属于关键门操作,但 g_2 不属于.如果执行了关键门操作 g_1,并将其从 DAG 中移除,g_3 就可被执行,首层门操作集合 F 也同时被更新.相比之下,优先处理 g_2 不会帮助更新首层门操作 F.

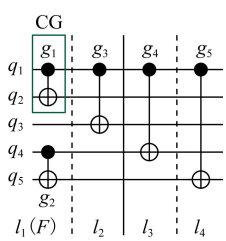


Fig. 10 Critical gates

图 10 关键门

对于每个程序 P_i,如果其首层门操作集合中存在满足映射位置约束条件可直接执行的门操作,则执行该门操作,将其从 DAG_i 中移除,并更新首层门操作集合.当所有程序的首层门操作集合中都没有

可以直接执行的门操作时,需要插入 SWAP,交换逻辑量子位的映射位置,将不可执行的门操作涉及的逻辑量子位移动到相邻的映射位置,从而使该门操作能被执行.为了快速更新首层门操作,缩小映射后线路深度,需要优先处理关键门操作.本文中的方法仅搜索与关键门操作涉及到的逻辑量子位相关的 SWAP 操作.

图 11 展示了一个量子程序映射转换的例子,其中 g_1 和 g_3 为关键门操作,所有与 g_1 和 g_3 中涉及到的逻辑量子位相关的 SWAP 操作被加入到候选 SWAP 操作中,通过启发式函数选出候选 SWAP 操作中的最优 SWAP 操作.当该 SWAP 操作被插入原线路后,量子程序映射被更新,一些原先不可执行的 CNOT 门操作可能变为可执行.重复迭代该过程,直到 DAG 中的所有 CNOT 操作都可执行.

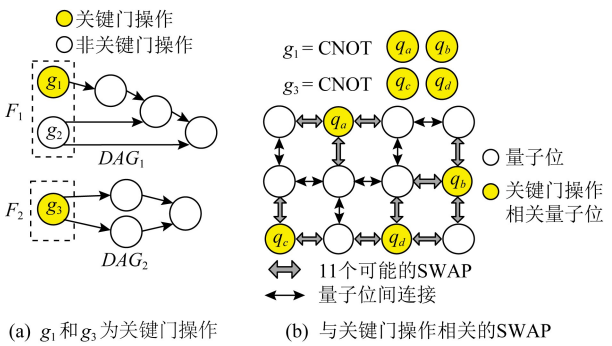


Fig. 11 SWAPs associated with critical gates

图 11 与关键门操作相关的 SWAP 操作

4.3 启发式成本函数设计

启发式函数负责从启发式搜索空间中所有的候选 SWAP 操作中找到最佳 SWAP.本文设计了新的启发式成本函数,利用并发量子程序映射问题的特点,鼓励高效跨程序 SWAP 操作的执行.

量子程序映射策略应使每个量子程序中各个量子位的映射位置相互靠近.否则,若 2 个量子位映射位置较远,如果要在这 2 个量子位之间执行 CNOT 操作,会引发很高的 SWAP 开销.SABRE[16]中使用了最近邻成本(nearest neighbor cost, NNC)函数来估算 SWAP 成本,进行量子程序映射.本文将 SABRE[16]使用的 NNC 函数 H 用作启发式函数的一部分.H 表示首层门操作集合的成本和扩展门操作集合的成本之和.扩展门操作集合是 DAG 中在首层门操作集合之后执行的 n 个(n 默认等于逻辑量子位数)CNOT 门操作的集合.每个门操作集合的成本计算为该集合中所有 CNOT 门的平均 SWAP 路径长度.

本文设计启发式函数,鼓励能够采取捷径的跨程序 SWAP 操作的执行.对于一个特定的量子计算机,已知其物理拓扑结构图和量子程序的初始映射,定义最短路矩阵 $\boldsymbol{D}$,矩阵中的每个元素代表量子计算机上任意 2 个物理量子位之间的最短路径长度.对于每个程序 P_i,定义 $\boldsymbol{D}'_i$ 为未被占用的物理量子位以及 P_i 映射到的物理量子位之间的最短路径矩阵.实质上,$\boldsymbol{D}$ 表示允许跨程序 SWAP 情况下对并发量子程序执行映射转换的最短路径矩阵.$\boldsymbol{D}'_i$ 表示对 P_i 独立执行映射转换,不允许跨程序 SWAP 操作时的最短路径矩阵.对于一个双量子位门操作 g,将 g 中涉及的 2 个逻辑量子位表示为 $g.q_1$ 和 $g.q_2$.将逻辑量子位 q 映射到的物理量子位定义为$\sigma(q)$.一个双量子位门操作中涉及的 2 个量子位的最短距离−1 是满足其约束所需的最小 SWAP 操作数.

在本文的设计中,对于量子程序 P_i 中的一个 CNOT 操作 g,$\boldsymbol{D}'_i[\sigma(g.q_1)][\sigma(g.q_2)]>\boldsymbol{D}[\sigma(g.q_1)][\sigma(g.q_2)]$代表在处理 g 的映射约束时,跨程序 SWAP 操作所需的成本低于程序内 SWAP 操作.这时,应当启用跨程序 SWAP 操作,节省映射转换成本.

例如,在图 9(b)中,由于需要 1 个跨程序 SWAP 或 3 个程序内 SWAP 来满足 CNOT q_1,q_5 的约束,所以 $\boldsymbol{D}[\sigma(q_1)][\sigma(q_5)]=2$,$\boldsymbol{D}'_i[\sigma(q_1)][\sigma(q_5)]=4$.本文定义跨程序 SWAP 策略相较于先前的策略带来的增益为

$$gain(g)=\boldsymbol{D}[\sigma(g.q_1)][\sigma(g.q_2)]-\boldsymbol{D}'_i[\sigma(g.q_1)][\sigma(g.q_2)]. \tag{2}$$

定义启发式函数为

$$score(SWAP)=H(SWAP)+\sum_{F_i\in F}\frac{1}{|F_i|}\sum_{g\in F_i}gain(g)I(SWAP,g). \tag{3}$$

因为不同首层门操作集合的大小不同,因此采用首层门操作的大小对 $gain$ 进行标准化.当 SWAP 处于门操作 g 的最短 SWAP 路径上时,指示函数 $I(SWAP,g)=1$,否则指示函数 $I(SWAP,g)=0$.在最短路上的 SWAP 操作被鼓励执行.候选 SWAP 中具有最小启发式函数值的 SWAP 操作是最优的 SWAP 操作.跨程序 SWAP 量子程序映射转换算法如算法 3 所示.

算法 3. 量子程序映射转换算法.

输入:量子芯片拓扑结构 G、量子程序 $programs$、初始映射 $mapping$;

输出:最终调度 $schedule$.

① 为每个程序生成一个 DAG;
② 为每个 DAG 生成一个首层门操作集合 F;
③ while 并非所有的 CNOT 门操作都可执行
④ if 在 F 中存在能够直接执行的门操作
⑤ 添加能直接执行的门操作到 $schedule$;
⑥ 将能直接执行的门操作从 DAG 中移除,更新 F;
⑦ else
⑧ for each F
⑨ 将F 中在第 2 层有邻接 CNOT 操作的 CNOT 操作添加入关键门操作集合 CG;
⑩ end for
⑪ 添加涉及到 CG 中逻辑量子位的 SWAP 操作到候选 SWAP 集合;
⑫ 从候选 SWAP 集合中选取 $score(SWAP)$最小的 $SWAP$;
⑬ 添加 $SWAP$ 到 $schedule$;
⑭ 更新 $mapping$;
⑮ end if
⑯ end while
⑰ return $schedule$.

5 映射任务调度器

尽管已有针对并发量子程序映射的机制[21],但选择并发量子程序组合仍有挑战性.现在的并发量子程序组合由用户指定,这可能导致 3 个问题:1)量子计算机资源未得到充分利用.2)随机选择的组合可能会导致保真度大幅降低.3)必须引入结果验证机制,这会带来额外的系统修改开销.本文提出了一种并发量子程序映射任务调度器的设计,用来选择合适的并发量子程序组合.图 12 展示了其工作流程.

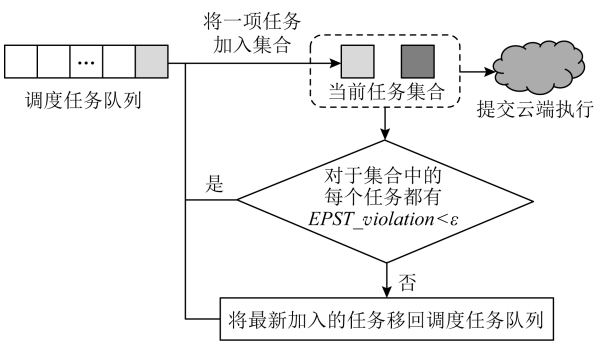


Fig. 12 The framework of the quantum program mapping task scheduler

图 12 量子程序映射任务调度器框架

我们的设计选择最优的量子程序组合,最大限度地提高了量子程序的执行保真度和量子计算机的资源利用率.对于调度队列首的任务,调度器检查任务队列中是否存在能够与队列首任务并发执行,且造成的保真度损失可被接受的其他量子程序.如果存在,则将它们与队列首任务同时映射到量子计算机上执行;否则,单独映射并执行队列首任务.

本文提出了预估实验成功率(estimated probability of a successful trial, $EPST$),用来估计在特定量子芯片上执行量子程序的保真度.Sep_EPST 表示程序单独映射时可以达到的最高 $EPST$.为了计算 Sep_EPST,需要对任务集合中的每个量子程序单独调用一次算法 2,为其分配一个物理量子位集合.而 Co_EPST 表示多个程序共同映射在量子芯片上时的 $EPST$.为了计算 Co_EPST,需要调用算法 2,为集合中所有的并发量子程序共同划分初始映射区域.$EPST$ 定义为

$$EPST=r_{2q}^{|\text{CNOTs}|}\ r_{1q}^{|\text{1q gates}|}\ r_{ro}^{|\text{qubits}|}, \tag{4}$$

其中,r_{2q},r_{1q}和 r_{ro}分别表示 CNOT、单量子位门操作和所分配的物理量子位的读出操作的平均可靠度.|CNOTs|,|1q gates|和|qubits|分别表示量子程序的 CNOT 门数、单量子位门数和逻辑量子位数.$EPST$ 高表示量子程序被映射到更健壮的区域,并且在实际执行期间程序的执行保真度更高.

对于特定的量子程序组合,调度器根据 Sep_EPST 和 Co_EPST 计算预估实验成功率损失 $EPST\_violation$.如果 $EPST\_violation$ 小于阈值 ε,则该量子程序组合可以并发执行.该调度器支持在 1 台量子计算机上并发执行 2 个以上的程序.为了确保调度器的效率和调度的公平性,仅搜索任务队列中的前 N 个任务(默认 $N=10$),限制并发程序数目不超过 M(默认 $M=3$).更多细节见算法 4.

算法 4. 映射任务调度算法.

输入:调度任务队列 J.

① while J 中仍有任务未被调度
② 初始化 cur_job_set 为仅包含 J 中第 1 项任务的列表;
③ $idx \leftarrow 1$;
④ while $idx<J.length$ 且 $idx<N$ 且 $cur\_job\_set.length<M$
⑤ $cur\_job\_set.append(J[idx])$;
⑥ for job in cur_job_set
⑦ 计算 job 的 Sep_EPST,Co_EPST;
⑧ 计算 job 的 $EPST\_violation=1-(Co\_EPST/Sep\_EPST)$;

⑨ if $EPST\_violation>\varepsilon$
⑩ $cur\_job\_set.remove(J[idx])$;
⑪ break;
⑫ end if
⑬ end for
⑭ $idx \leftarrow idx+1$;
⑮ end while
⑯ 调用算法 3 映射 cur_job_set 中的任务,提交执行;
⑰ 从 J 中移除 cur_job_set 中的任务;
⑱ end while

6 实验评测

6.1 评测方法

1) 评测指标

① 实验成功率(probability of a successful trial, *PST*).*PST* 用于评估量子程序执行保真度,*PST* 被定义为实验中能够产生正确结果的执行次数占总执行次数的比例.我们在量子计算机上对每个工作负载进行映射,执行 8 024 次,统计 *PST*.

② 映射后量子门数量.我们使用映射后 CNOT 门操作的数量来评估映射策略在映射并发量子程序时缩减映射成本的能力.

③ 映射后量子线路深度.我们使用映射后量子程序的线路深度评估映射策略缓减保持性错误的能力.

④ 执行次数减小因数(trial reduction factor, *TRF*).*TRF* 用于评估并发量子程序映射策略带来的通量的提升.*TRF* 被定义为独立执行多个程序时所需的执行次数与启用并发量子程序映射时所需的执行次数之比.

2) 量子芯片

我们在 IBM-Q16[11] 和 IBM-Q50[11] 上进行评估,它们的量子位拓扑结构分别如图 2 和图 13 所示,其中 IBM-Q16 可通过 IBM 接口[11] 访问.而 IBM-Q50 并不公开,我们模拟了 IBM-Q50 芯片用于映射成本评估.模拟芯片包括拓扑信息和错误率数据.我们使用均匀分布随机模型,在 IBM-Q16 每个属性最大值和最小值的范围内生成了 IBM-Q50 的错误率数据.

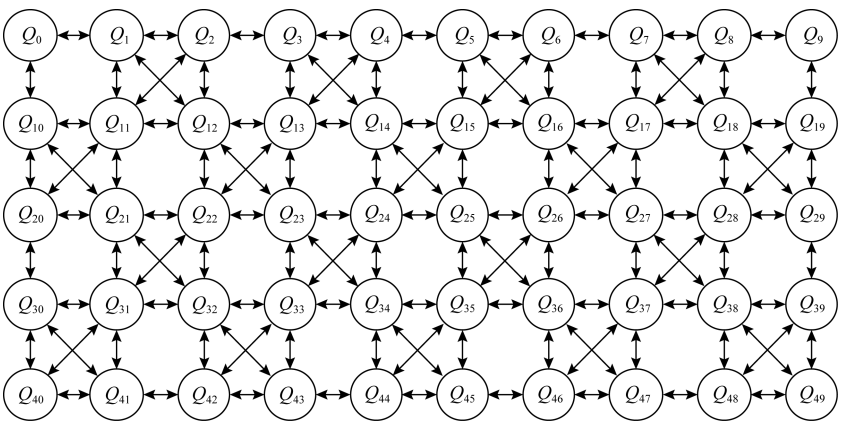


Fig. 13 Qubit topology of IBM-Q50
图 13 IBM-Q50 的量子位拓扑结构

3) 数据集

我们采用先前研究中使用的数据集,包括 SABRE[16], QASM-Bench[55], RevLib[56] 和文献[50]中使用的数据集.如表 3 所示,微型/小型量子程序大约有 5 个量子位和数十个 CNOT 门;大型程序大约有 10 个量子位和数百个 CNOT 门.

Table 3 Benchmarks Used in the Evaluation
表 3 实验评测中所用的工作集

类型	工作集
微型	bv_n3,bv_n4,peres_3,toffoli_3,fredkin_3
小型	3_17_13,decod24-v2_43,4mod5-v1_22,mod5mils_65,alu-v0_27
大型	aj-e11_165,4gt4-v0_72,alu-bdd_288,ex2_227,ham7_104,bv_n10,ising_model_10,qft_10,sys6-v0_111,rd53_311,qft_16,alu-v2_31,C17_204,cnt3-5_180,sf_276,sym9_146

4) 对比实验

① 单独执行.使用 Qiskit[26] 中优化级别最高的算法分别映射和执行工作负载中的每个程序.它们代表最可靠的情况,没有并发程序之间的干扰.

② 并发基准.采用文献[21]中提出的策略,用

FRP 策略生成并发量子程序的初始映射,用引进基于错误率的优化的 SABRE 进行映射转换.

③ SABRE[16].SABRE 是一种以 SWAP 操作数为优化目标的启发式映射方法.该策略被用于映射多个并发量子程序合并成的量子线路.

本文中的策略被细分为:仅使用 CDAQP、仅使用 X-SWAP,以及同时使用 CDAQP 和 X-SWAP.其中,仅使用 CDAQP 的方法采用了与 SABRE 相同的映射转换方法,仅使用 X-SWAP 的方法使用了与 SABRE 相同的初始映射策略.

6.2 程序执行保真度评测

本文使用微型/小型量子程序组合进行保真度评测.表 4 展示了在 IBM-Q16 上执行的包含 2 个程序的工作负载的 *PST*.相同程序组合下的实验均在 IBM-Q16 的一个错误率标定周期内完成,即量子计算机的错误率标定数据相同.量子程序的两两组合可以使量子计算机的通量提高一倍.但是,由于资源冲突和串扰,量子程序的执行可靠性可能会降低.在大多数情况下,并发量子程序映射的平均 *PST* 低于单独执行情况下的平均 *PST*.但本文的方法仍优于其他并发量子程序映射策略.对于微型程序组合,本文的方法(CDAQP + X-SWAP)、单独执行、SABRE 和并发基准的平均 *PST* 分别为 58.2%,69.8%,44.8%和 45.3%;对于小型工作负载,相应的平均 *PST* 分别为 16.8%,23.8%,10.3%和 12.4%.本文方法的执行保真度比 SABRE 和并发基准分别平均高出 9.9%和 8.6%.这表明本文的方法提供了更可靠的结果,并且减少了保真度的损失.

Table 4 *PST* Comparison Between Multiple Strategies on IBM-Q16

表 4 IBM-Q16 上多种策略 *PST* 的对比 %

程序组合		单独执行			SABRE			并发基准			CDAQP+X-SWAP			仅 CDAQP			仅 X-SWAP		
W1	W2	PST_1	PST_2	平均	PST_1	PST_2	平均	PST_1	PST_2	平均	PST_1	PST_2	平均	PST_1	PST_2	平均	PST_1	PST_2	平均
bv_n3	bv_n3	83.4	84.0	83.7	50.8	75.4	63.1	57.5	61.4	59.5	69.7	66.2	68.0	68.7	67.1	67.9	19.3	54.9	37.1
bv_n3	bv_n4	83.4	63.6	73.5	52.6	34.5	43.5	35.9	25.0	30.4	52.5	17.7	35.1	27.2	12.8	20.0	37.4	53.7	45.5
bv_n3	peres_3	83.9	81.2	82.5	56.3	51.2	53.8	49.3	61.0	55.2	63.9	71.1	67.5	65.1	70.8	67.9	67.9	76.8	72.4
bv_n3	toffoli_3	83.1	82.0	82.5	59.0	49.5	54.3	65.7	41.9	53.8	65.3	81.7	73.5	63.8	70.5	67.1	48.5	76.5	62.5
bv_n3	fredkin_3	83.2	46.2	64.7	40.9	56.1	48.5	69.7	39.1	54.4	66.1	81.9	74.0	64.9	82.1	73.5	46.4	64.1	55.2
bv_n4	bv_n4	50.7	51.3	51.0	30.4	51.8	41.1	49.2	28.7	39.0	51.1	52.8	52.0	48.5	37.7	43.1	49.6	41.6	45.6
peres_3	peres_3	63.1	59.0	61.1	41.4	20.3	30.9	45.7	46.9	46.3	51.2	54.4	52.8	51.1	52.9	52.0	44.0	13.2	28.6
toffoli_3	toffoli_3	64.3	65.5	64.9	21.9	33.7	27.8	22.8	33.6	28.2	50.2	51.0	50.6	52.4	41.3	46.9	39.1	51.2	45.2
fredkin_3	fredkin_3	65.4	63.0	64.2	39.6	40.0	39.8	40.9	41.3	41.1	52.0	48.1	50.1	48.5	22.1	35.3	52.3	42.0	47.1
平均				69.8			44.8			45.3			58.2			52.6			48.8
3_17_13	3_17_13	43.1	44.6	43.8	12.9	10.8	11.9	14.0	11.6	12.8	33.3	11.3	22.3	7.0	22.2	14.6	23.0	15.6	19.3
3_17_13	4mod5-v1_22	45.0	28.3	36.7	10.4	18.7	14.5	12.0	21.3	16.7	33.5	17.3	25.4	13.7	29.6	21.7	16.8	25.4	21.1
3_17_13	mod5mils_65	22.4	29.2	25.8	9.3	3.6	6.5	18.5	19.1	18.8	32.0	16.9	24.5	13.7	7.6	10.7	13.2	16.7	15.0
3_17_13	alu-v0_27	44.0	14.1	29.0	9.0	7.3	8.1	14.0	7.4	10.7	18.3	15.2	16.8	20.9	21.9	21.4	9.3	6.3	7.8
3_17_13	decod24-v2_43	43.6	6.2	24.9	18.2	7.8	13.0	11.5	9.3	10.4	14.7	11.2	13.0	27.2	6.6	16.9	7.6	19.1	13.4
4mod5-v1_22	4mod5-v1_22	29.2	26.0	27.6	13.4	21.0	17.2	17.8	20.3	19.1	16.4	22.3	19.4	18.9	21.3	20.1	19.5	16.2	17.8
mod5mils_65	mod5mils_65	11.6	8.6	10.1	6.8	9.3	8.1	4.1	14.7	9.4	8.7	12.0	10.4	7.5	7.7	7.6	7.8	10.2	9.0
alu-v0_27	alu-v0_27	9.1	10.0	9.6	9.9	3.4	6.7	8.8	6.6	7.7	8.8	12.9	10.8	7.7	10.5	9.1	5.1	5.0	5.0
decod24-v2_43	decod24-v2_43	6.0	6.8	6.4	6.9	6.1	6.5	7.7	5.3	6.5	7.5	9.4	8.4	10.1	6.2	8.1	7.2	7.6	7.4
平均				23.8			10.3			12.4			16.8			14.5			12.9

本文的方法主要受益于 CDAQP 策略.CDAQP 提供了更好的初始映射来提高保真度,这使得并发量子程序执行时的保真度接近甚至超过单独执行情况下的保真度.如表 4 所示,在 bv_n3 和 fredkin_3 组合中,仅使用 CDAQP 的策略通过提供更好的初始映射,使保真度比并发基准显著提高了 19.1%.

CDAQP 相对于并发基准的优势来自 2 个方面:1)在可靠量子位和连接上运行的量子门具有较低的错误率.2)较好的初始分配降低了映射转换时的 SWAP 开销.总体上,仅使用 CDAQP 的策略减少了并发量子程序映射带来的保真度下降,保真度相较于并发基准高出 4.7%.

仅使用 X-SWAP 的策略在提升保真度方面没有体现出显著优势,这是因为:1)映射小型量子程序需要很少的 SWAP 操作,因此 X-SWAP 在这种情况下很难节省 SWAP.2)在 IBM-Q16 这种结构简单的量子芯片上,初始映射对小型量子程序的可靠性有很大影响.3)量子程序映射位置可能不相邻,难以执行跨程序 SWAP.但是,X-SWAP 在具有更多量子位的芯片上却能有较好的表现.

6.3 映射成本评测

X-SWAP 在更大的启发式搜索空间中表现更好.当多个更大规模的量子程序在具有更多量子位的量子芯片上并发运行时,X-SWAP 能体现出更优的性能.本文将映射后量子门操作的数量和线路深度作为评价标准,评估 IBM-Q50 上 4 程序工作负载的映射开销.工作负载随机选择,旨在覆盖尽可能多的正交程序组合.对于每个工作负载和每个策略,本文采用 5 次实验中的最佳结果(类似于文献[16]).实验结果如表 5 所示.

SABRE[16]使用反向遍历技术和启发式搜索方案,最小化插入 SWAP 的数量.然而,因 SABRE 在初始映射时并未考虑局部性原理,在映射并发量子程序工作负载时,并发量子程序的 SWAP 路径存在交叠,引发了更多的 SWAP 操作,无法获得最优映射方案.并发基准在划分量子位时会考虑局部性,但在并发量子程序之间会引入资源冲突,导致冗余的 SWAP 操作.实验结果表明,并发基准的映射后 CNOT 门操作数平均比 SABRE 高 4.0%,映射后线路深度平均比 SABRE 高 6.8%.

Table 5 Mapping Overheads Comparison of 4-program Workloads on IBM-Q50

表 5 IBM-Q50 上的 4 程序工作集映射成本对比

组合	工作负载中的量子程序	SABRE		并发基准		CDAQP+X-SWAP		仅 CDAQP		仅 X-SWAP	
		CNOTs	深度	CNOTs	深度	CNOTs	深度	CNOTs	深度	CNOTs	深度
Mix_1	aj-e11_165,alu-v2_31,4gt4-v0_72,sf_276	1386	768	1303	847	1167	586	1141	530	1177	629
Mix_2	alu-bdd_288,ex2_227,ham7_104,C17_204	1222	596	1266	717	1236	671	1225	581	1242	717
Mix_3	bv_n10,ising_model_10,qft_10,sys6-v0_111	387	189	382	178	351	160	419	236	341	224
Mix_4	aj-e11_165,alu-v2_31,ising_model_10,cnt3-5_180	979	450	1008	437	953	480	1005	450	950	496
Mix_5	4gt4-v0_72,sf_276,sym9_146,rd53_311	1365	672	1429	759	1255	536	1237	497	1089	450
Mix_6	alu-bdd_288,ex2_227,qft_10,sys6-v0_111	871	711	862	655	780	563	862	578	809	579
Mix_7	ham7_104,C17_204,bv_n10,ising_model_10	713	370	963	496	668	413	723	423	677	399
Mix_8	aj-e11_165,4gt4-v0_72,rd53_311,cnt3-5_180	996	448	1052	537	916	441	894	368	940	390
Mix_9	alu-v2_31,sf_276,sym9_146,qft_16	1481	738	1499	769	1478	718	1545	779	1325	482
Mix_10	alu-bdd_288,ham7_104,ising_model_10,sys6-v0_111	659	392	662	315	622	248	656	306	617	325
Mix_11	ex2_227,C17_204,bv_n10,qft_10	955	537	1049	630	900	569	964	531	985	599
Mix_12	aj-e11_165,sf_276,C17_204,sys6-v0_111	1438	834	1337	828	984	506	1267	773	1063	467

平均而言,仅使用 CDAQP 在映射后 CNOT 门的数量上比 SABRE 高 2.7%,在线路深度上比 SABRE 高 6.8%.虽然 CDAQP 能够帮助找到量子位紧密相连的初始映射.帮助减小映射开销,但效果并不显著.在个别情况下(如 Mix_9),仅使用 CDAQP 反而造成了比其他策略还高的映射开销.其原因是:CDAQP 的主要优化目标是提升初始映射的保真度.降低映射开销(即用最少的 SWAP 完成量子程序映射)并不是 CDAQP 的首要任务.仅采用 X-SWAP 的策略使用了与 SABRE 相同的初始映射方法,允许执行跨程序 SWAP 操作,优先处理关键门操作,有效地将映射后门操作的数量缩减了 8.8%,并将映射后线路深度缩减了 9.2%.原因包括 2 个方面:1)X-SWAP 利用跨程序 SWAP,采取捷径,节省映射开销.2)通过优先处理关键门操作,X-SWAP 可提供更高效的 SWAP,缩减映射开销和

映射后线路深度.

在本文的设计中,CDAQP 可以生成可靠且紧密相连的初始映射,而 X-SWAP 则用于减少映射开销.同时应用 CDAQP 和 X-SWAP 可提升性能:与并发基准和 SABRE 相比,映射后门操作数分别减少了 11.6%和 8.6%;与并发基准和 SABRE 相比,线路深度分别降低了 16.0%和 10.3%.表 5 展示了更详细的数据.

此外,本工作具有可扩展性。本工作不仅能够降低 IBM-Q50 上的 4 程序工作负载的映射开销,还能在更大规模的量子芯片上运行.这是因为:1)CDAQP 中所采用的社区检测方法已被证明对于大型网络是有效的.2)无论量子芯片规模大小,只要量子程序映射位置邻接,X-SWAP 就可以起到减少映射开销的作用.

6.4 任务调度器评测

本文采用表 3 中的微型/小型量子程序构造了一个映射任务队列,并使用任务调度器进行调度.从 0.05～0.20 变更阈值 ε,在每种情况下分别进行调度,工作负载的 *PST* 均值和 *TRF* 如图 14 所示.图 14 还展示了在每个程序单独执行和程序两两随机组合情况下的性能.

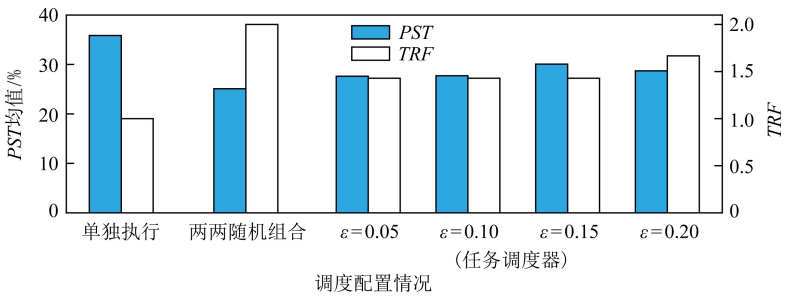


Fig. 14 Performance of the task compiler

图 14 任务调度器性能

单独执行情况下,*TRF* 为 1(无并发),平均 *PST* 为 35.9%,是该任务队列能够达到的最高平均 *PST*.程序两两随机组合情况下的平均 *PST* 最低,仅为 25.1%,但 *TRF* 达到 2(2 个程序并发).当 $\varepsilon=0.15$(即仅允许 *EPST* 损失小于 15%的并发量子程序映射情况)时,调度器性能最好.在这种情况下,工作负载的平均保真度为 30.1%,比两两随机组合的工作负载高出 5.0%;*TRF* 为 1.429,与单独执行的情况相比,吞吐量提高了 42.9%.实验结果表明,本文的映射任务调度器可以用于权衡量子计算机的吞吐量和保真度.

7 总 结

量子计算和量子计算云服务逐渐兴起.当前云环境下的量子计算机面临着资源利用率低、保真度低、错误率高等问题.本文对面向超导量子芯片的量子程序映射技术进行了综述,深入剖析了不同类别映射技术的特点和区别.并针对云环境下的并发量子程序映射问题提出了一种新的映射策略,提升并发量子程序的执行保真度和资源利用率.本文设计的系统是一个面向量子计算机的操作系统原型 QuOS.希望我们的工作能帮助相关领域未来的研究人员.

参 考 文 献

[1] Grover L K. A fast quantum mechanical algorithm for database search [C] //Proc of the 28th Annual ACM Symp on Theory of Computing. New York: ACM, 1996: 212-219

[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 [J]. arXiv preprint arXiv:1307.0411, 2013

[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(1): 4213

[6] Wang Baonan, Hu Feng, Zhang Huanguo, et al. From evolutionary cryptography to quantum artificial intelligent cryptography [J]. Journal of Computer Research and Development, 2019, 56(10): 2112-2134 (in Chinese)
(王宝楠, 胡风, 张焕国, 等. 从演化密码到量子人工智能密码综述[J]. 计算机研究与发展, 2019, 56(10): 2112-2134)

[7] Du Weilin, Li Bin, Tian Yu. Quantum annealing algorithms: State of the art [J]. Journal of Computer Research and Development, 2008, 45(9): 1501-1508 (in Chinese)
(杜卫林, 李斌, 田宇. 量子退火算法研究进展[J]. 计算机研究与发展, 2008, 45(9): 1501-1508)

[8] Hsu J. CES 2018: Intel's 49-Qubit Chip Shoots for Quantum Supremacy [EB/OL]. [2021-03-26]. https://spectrum.ieee.org/tech-talk/computing/hardware/intels-49qubit-chip-aims-for-quantum-supremacy

[9] Kelly J. A Preview of Bristlecone, Google's New Quantum Processor [EB/OL]. [2021-03-26]. https://ai.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html

[10] Knight W. IBM Raises the Bar with a 50-Qubit Quantum Computer [EB/OL]. [2021-03-26]. https://www.technologyreview.com/2017/11/10/147728/ibm-raises-the-bar-with-a-50-qubit-quantum-computer/

[11] IBM. IBM Quantum Experience [CP/OL]. [2021-03-26]. https://quantum-computing.ibm.com/

[12] Preskill J. Quantum computing in the NISQ era and beyond [J]. Quantum, 2018, 2: No.79

[13] Koch J, Terri M Y, Gambetta J, et al. Charge-insensitive qubit design derived from the Cooper pair box [J]. Physical Review A, 2007, 76(4): 042319

[14] 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

[15] Zhong Hansen, Wang Hui, Deng Yuhao, et al. Quantum computational advantage using photons [J]. Science, 2020, 370(6523): 1460-1463

[16] Li Gushu, Ding Yufei, Xie Yuan. Tackling the qubit mapping problem for NISQ-era quantum devices [C] //Proc of the 24th Int Conf on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2019: 1001-1014

[17] Ash-Saki A, Alam M, Ghosh S. QURE: Qubit re-allocation in noisy intermediate-scale quantum computers [C] //Proc of the 56th Annual Design Automation Conf. New York: ACM, 2019: 1-6

[18] Murali P, Baker J M, Javadi-Abhari A, et al. Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers [C] //Proc of the 24th Int Conf on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2019: 1015-1029

[19] Tannu S S, Qureshi M K. Not all qubits are created equal [C] //Proc of the 24th Int Conf on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2019: 987-999

[20] Murali P, Mckay D C, Martonosi M, et al. Software mitigation of crosstalk on noisy intermediate-scale quantum computers [C] //Proc of the 25th Int Conf on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2020: 1001-1016

[21] Das P, Tannu S S, Nair P J, et al. A case for multi-programming quantum computers [C] //Proc of the 52nd Annual IEEE/ACM Int Symp on Microarchitecture. New York: ACM, 2019: 291-303

[22] Barenco A, Bennett C H, Cleve R, et al. Elementary gates for quantum computation [J]. Physical Review A, 1995, 52(5): 3457-3467

[23] 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

[24] Chong F T, Franklin D, Martonosi M. Programming languages and compiler design for realistic quantum hardware [J]. Nature, 2017, 549(7671): 180-187

[25] Tucci R R. A rudimentary quantum compiler [J]. arXiv preprint quant-ph/9805015, 1998

[26] Abraham H, Offei-danso A, Agarwal R, et al. Qiskit: An Open-source Framework for Quantum Computing [CP/OL]. [2021-03-26]. https://github.com/Qiskit/qiskit

[27] Zulehner A, Wille R. Compiling SU(4) quantum circuits to IBM QX architectures [C] //Proc of the 24th Asia and South Pacific Design Automation Conf. New York: ACM, 2019: 185-190

[28] Huang Yipeng, Martonosi M. Statistical assertions for validating patterns and finding bugs in quantum programs [C] //Proc of the 46th Int Symp on Computer Architecture. New York: ACM, 2019: 541-553

[29] Liu Ji, Zhou Huiyang. Systematic approaches for precise and approximate quantum state runtime assertion [C] //Proc of the 27th IEEE Int Symp on High Performance Computer Architecture. Piscataway, NJ: IEEE, 2021: 179-193

[30] Travis L, Fang Qi, Lu Peng. Robust cache-aware quantum processor layout [C] //Proc of the 39th Int Symp on Reliable Distributed Systems. Piscataway, NJ: IEEE, 2020: 276-287

[31] Dou Xinglei, Liu Lei. A new qubits mapping mechanism for multi-programming quantum computing [C] //Proc of the 29th Int Conf on Parallel Architectures and Compilation Techniques. New York: ACM, 2020: 349-350

[32] Liu Lei, Dou Xinglei. QuCloud: A new qubit mapping mechanism for multi-programming quantum computing in cloud environment [C] //Proc of the 27th IEEE Int Symp on High Performance Computer Architecture. Piscataway, NJ: IEEE, 2021: 167-178

[33] Siraichi M Y, Santos V F D, Collange S, et al. Qubit allocation [C] //Proc of the 2018 Int Symp on Code Generation and Optimization. New York: ACM, 2018: 113-125

[34] Tan Bochen, Cong J. Optimality study of existing quantum computing layout synthesis tools [J]. arXiv preprint arXiv: 2002.09783v4, 2020

[35] 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

[36] Shafaei A, Saeedi M, Pedram M. Qubit placement to minimize communication overhead in 2D quantum architectures [C] //Proc of the 19th Asia and South Pacific Design Automation Conf. Piscataway, NJ: IEEE, 2014: 495-500

[37] Wille R, Lye A, Drechsler R. Optimal SWAP gate insertion for nearest neighbor quantum circuits [C] //Proc of the 19th Asia and South Pacific Design Automation Conf. Piscataway, NJ: IEEE, 2014: 489-494

[38] Lye A, Wille R, Drechsler R. Determining the minimal number of swap gates for multi-dimensional nearest neighbor quantum circuits [C] //Proc of the 20th Asia and South Pacific Design Automation Conf. Piscataway, NJ: IEEE, 2015: 178-183

[39] Booth K, Do M, Beck J, et al. Comparing and integrating constraint programming and temporal planning for quantum circuit compilation [C] //Proc of the 28th Int Conf on Automated Planning and Scheduling. Palo Alto, CA: AAAI, 2018: 366-374

[40] Bhattacharjee D, Saki A A, Alam M, et al. MUQUT: Multi-constraint quantum circuit mapping on NISQ computers [C] // Proc of the 38th IEEE/ACM Int Conf on Computer-Aided Design. Piscataway, NJ: IEEE, 2019: 1-7

[41] Wille R, Burgholzer L, Zulehner A. Mapping quantum circuits to IBM QX architectures using the minimal number of SWAP and H operations [C] //Proc of the 56th ACM/IEEE Design Automation Conf. Piscataway, NJ: IEEE, 2019: 1-6

[42] Murali P, Linke N M, Martonosi M, et al. Full-stack, real-system quantum computer studies [C] //Proc of the 46th Int Symp on Computer Architecture. New York: ACM, 2019: 527-540

[43] Tan Bochen, Cong J. Optimal layout synthesis for quantum computing [C] //Proc of the 39th IEEE/ACM Int Conf on Computer Aided Design. New York: ACM, 2020: 1-9

[44] Gurobi. Gurobi Optimizer Reference Manual [EB/OL]. [2021-03-26]. https://www.gurobi.com/documentation/9.1/refman/index.html

[45] De Moura L, Bjørner N. Z3: An efficient SMT solver [C] // Proc of the 11th Int Conf on Tools and Algorithms for the Construction and Analysis of Systems. Berlin: Springer, 2008: 337-340

[46] Bjørner N, Phan A-D, Fleckenstein L. vZ—An optimizing SMT solver [C] //Proc of the 18th Int Conf on Tools and Algorithms for the Construction and Analysis of Systems. Berlin: Springer, 2015: 194-199

[47] Shafaei A, Saeedi M, Pedram M. Optimization of quantum circuits for interaction distance in linear nearest neighbor architectures [C] //Proc of the 50th ACM/EDAC/IEEE Design Automation Conf. New York: ACM, 2013: 1-6

[48] Wille R, Keszocze O, Walter M, et al. Look-ahead schemes for nearest neighbor optimization of 1D and 2D quantum circuits [C] //Proc of the 21st Asia and South Pacific Design Automation Conf. Piscataway, NJ: IEEE, 2016: 292-297

[49] Bhattacharjee A, Bandyopadhyay C, Wille R, et al. A novel approach for nearest neighbor realization of 2D quantum circuits [C] //Proc of the 2018 IEEE Computer Society Annual Symp on VLSI. Piscataway, NJ: IEEE, 2018: 305-310

[50] Zulehner A, Paler A, Wille R. Efficient mapping of quantum circuits to the IBM QX architectures [C] //Proc of the 2018 Design, Automation Test in Europe Conf Exhibition. Piscataway, NJ: IEEE, 2018:1135-1138

[51] Smith K N, Thornton M A. A quantum computational compiler and design tool for technology-specific targets [C] // Proc of the 46th Int Symp on Computer Architecture. New York: ACM, 2019: 579-588

[52] Siraichi M Y, Santos V F D, 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(OOPSLA): 1-29

[53] Ding Yongshan, Wu Xinchuan, Holmes A, et al. SQUARE: Strategic quantum ancilla reuse for modular quantum programs via cost-effective uncomputation [C] //Proc of the 47th ACM/IEEE Annual Int Symp on Computer Architecture. Piscataway, NJ: IEEE, 2020: 570-583

[54] Newman M E. Fast algorithm for detecting community structure in networks [J]. Physical Review E, 2004, 69(6): 066133

[55] Cross A W, Bishop L S, Smolin J A, et al. Open quantum assembly language [J]. arXiv preprint arXiv: 1707.03429, 2017

[56] Wille R, Große D, Teuber L, et al. RevLib: An online resource for reversible functions and reversible circuits [C] // Proc of the 38th Int Symp on Multiple Valued Logic. Piscataway, NJ: IEEE, 2008: 220-225

Dou Xinglei, born in 1998. Master candidate of Sys-Inventor Lab. His main research interests include computer architecture, quantum computer system, and operating system.

窦星磊, 1998 年生. Sys-Inventor Lab 硕士研究生. 主要研究方向为计算机体系结构、量子计算机系统、操作系统.

Liu Lei, born in 1981. PhD, associate professor, master supervisor, PI of Sys-Inventor Lab. His main research interests include computer architecture, new memory system, new operating system, and quantum computer system.

刘 磊, 1981 年生. 博士, 副研究员, 硕士生导师. Sys-Inventor Lab 负责人. 主要研究方向是计算机体系结构、新内存系统、新操作系统、量子计算机系统.

Chen Yuetao, born in 1999. Master candidate of Sys-Inventor Lab. His main research interests include computer architecture, quantum computer system and operating system.

陈岳涛, 1999 年生. Sys-Inventor Lab 硕士研究生. 主要研究方向为计算机体系结构、量子计算机系统、操作系统.