Q-Profile：应用于量子近似优化算法的量子控制堆栈性能分析工具

Koen J. Mesman*Qblox*代尔夫特，荷兰
koenmesman@gmail.com

Francesco Battistel Edgar Reehuis*Qblox*
Qblox 代尔夫特，荷兰 代尔夫特，荷兰
fbattistel@qblox.com
ereehuis@qblox.com

Damaz de Jong*Qblox*代尔夫特，荷兰
ddejong@qblox.com

Marijn J. Tiggelman*Qblox*代尔夫特，荷兰
mtiggelman@qblox.com

Jordy Gloudemans*Qblox*代尔夫特，荷兰
jgloudemans@qblox.com

Jules C. van Oven*Qblox*代尔夫特，荷兰
jules@qblox.com

Cornelis C. Bultink*Qblox*代尔夫特，荷兰
niels@qblox.com

arXiv:2303.01450v1 [quant-ph] 2 Mar 2023

摘要—**增加量子比特数量并加快量子算法的执行，是实现量子优势的重要步骤。这对控制堆栈提出了很高的要求，尤其如此：脉冲需要分发至数量不断增加的控制通道，而变分算法则要求量子计算与经典计算快速交替进行。因此，评估控制堆栈中的瓶颈，是使其为实现量子优势做好准备的关键。然而，现有基准测试套件由于无法直接访问控制硬件，提供的细节不足。在本研究中，我们介绍了 Q-Profile，这是一种用于对量子控制堆栈进行性能分析的工具。它通过在主机 CPU 与控制堆栈之间建立直接连接，避免了上述问题，从而能够精确测量运行时间并识别性能瓶颈。我们通过在 Qblox Cluster 上对量子近似优化算法（量子近似优化算法）进行基准测试，展示了该工具的用法；测试对象是一个虚拟的 4 至 14 量子比特 transmon 系统。结果找出了主要的执行瓶颈：被动量子比特重置和通信开销。我们估计，与总运行时间相比，采用主动量子比特重置而非被动复位可实现 1.40 倍的加速；通过并行初始化控制模块，还可进一步实现 1.37 倍的加速。所介绍的性能分析方法适用于其他控制堆栈提供商，也适用于其他基准测试，同时仍能提供超越单一指标的详细信息。进一步而言，该工具将有助于识别并消除瓶颈，为未来的量子加速铺平道路。该性能分析工具包含在开源量子控制软件 Quantify 中，可支持多个后端。**

索引词—**量子控制堆栈、含噪中等规模量子、性能分析、基准测试、加速**

I. 引言

近年来，量子计算发展迅速，正朝着比任何经典计算机更快地执行计算任务的目标迈进 [1]。如今，人们寻求的下一个里程碑，是实现具有实际应用价值的快速计算，这被称为量子优势。要实现这一目标，受控量子比特的数量必须扩大，门错误率必须降低，二者都要改变多个数量级。这对量子控制栈（包括硬件和软件）提出了很高的要求：程序和脉冲需要分发到越来越多的控制通道，而混合优化算法 [2], [3]和量子门调谐协议 [4] 则要求快速交替进行量子电路执行、分析和经典计算。 此外，先前的结果 [5]表明，加速变分量子算法十分重要，因为针对最简单的示例问题（5 个量子比特），测得的最坏情况下运行时间超过 15 小时。因此，评估并了解控制堆栈中的瓶颈，是使其具备实现量子优势能力的关键。过去几年中，人们开发了多种基准测试套件 [5]–[8], ，但由于无法直接访问和连接控制硬件，这些套件存在精度不足的问题。

在本文中，我们介绍了 Q-Profile，这是一种用于对量子控制堆栈进行基准测试的工具。该工具通过利用主机 CPU 与量子控制堆栈之间的直接连接，规避了长距离、高延迟连接。借助对已调度指令的控制硬件运行时间的访问，该工具能够以精细的粒度和高准确度测量量子运行时间。这种性能分析方法旨在带来实际效益，使量子控制栈开发人员能够在其系统上执行近期量子算法时识别性能瓶颈。

为了展示 Q-Profile，我们采用了 QPack 基准测试套件中的量子近似优化算法（量子近似优化算法） [2]实现 [5], ，该套件是一个量子应用基准测试套件。我们使用这一实现来展示量子控制硬件的性能分析方法，而非提供基准测试分数；演示对象是用于 4 至 14 个量子比特虚拟传输子系统的 Qblox Cluster。这意味着 Cluster 的运行方式仿佛连接着实际的传输子量子比特，但仅对 QAOA 优化器在主机 CPU 上使用的二进制测量结果进行模拟。QPack套件

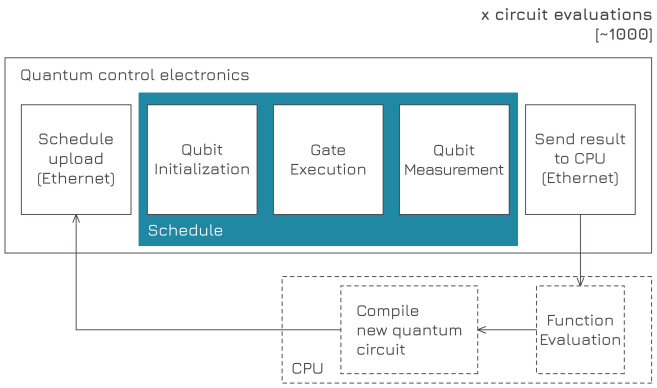

图 1. 量子计算的一般布局。主机 CPU 将量子电路编译为可执行的操作调度。该调度随后上传至量子控制硬件（例如通过以太网），由后者向量子比特发送适当的脉冲，以执行各项操作（初始化、量子门、测量）。通常会重复运行该电路多次以收集统计数据，最终将结果发送回主机 CPU 进行后处理。

此前的研究曾使用该套件对 IBM 硬件进行性能分析 [5]，但由于无法充分访问底层硬件，结果并不理想。相反，直接访问控制硬件后，云通信和排队时间都不复存在，大大加快了性能分析流程。我们的研究发现，主要的执行瓶颈在于量子比特重置模式（被动与主动）以及通信开销。针对这些瓶颈，我们提出并实施了缓解策略：与基线测量相比，主动复位实现了 1.40 倍的加速；通过并行模块初始化，又进一步实现了 1.37 倍的加速。这种性能分析方法适用于其他控制堆栈提供商以及其他基准测试。该性能分析工具已纳入开源的 Quantify [9] 量子控制软件中，可支持多个后端。此外，Q-Profile 还可用于测量电路层操作每秒数（CLOPS） [10]：运行类似量子体积的电路，而非 QAOA 电路，同时仍能提供单一指标之外的详细信息。通过对计算负荷最大的计算步骤进行性能分析，开发者能够高效地定位并缓解软件和硬件中的性能问题。进一步而言，该工具将有助于发现并消除未来量子加速领域中的瓶颈。

我们简要介绍量子硬件结构（II）、用于评估硬件的量子算法（III）以及所用指标（IV）。随后介绍性能测量结果（V）并对其进行讨论（VI），其中包括提升性能的潜在策略。第 VII 节总结了我们的研究发现。

II. 量子控制硬件结构

不同实现方式的量子计算机之间可能存在显著差异。尤其对于量子计算芯片，目前尚未就哪种技术将成为标准达成共识 {v1}。

然而，就量子计算控制硬件结构而言，仍可发现许多相似之处。执行量子计算的一般结构如图 1 所示。通用计算、预处理和后处理需要一台主机经典 CPU。主机 CPU 将高级量子算法编译为可执行的量子电路指令。这些指令会传输至量子控制硬件，例如通过以太网传输。控制硬件根据量子芯片技术的不同，以不同方式执行量子门和测量。控制硬件通常会存储、读取并解析接收到的指令，以生成相应的脉冲序列。此外，测量结果需要先被存储、读取，并可选择性地进行处理，然后再将结果发送回主机 CPU。量子电路的一次执行若包含最终测量，则称为一个实验轮次。为了收集准确的统计数据，通常会针对量子电路的每个实例执行数千次甚至更多实验轮次。测量结果要么在每个实验轮次结束时传输回去，要么暂时保存在本地，从而降低延迟。

III. 混合量子算法

在当前的量子计算时代，即所谓的 NISQ时代（含噪中等规模量子时代），量子比特在数量和质量上都有限，这限制了当前量子计算机能够解决的问题范围。诸如著名的 Shor因数分解算法或 Grover搜索算法等算法，需要大量量子比特才能计算实际应用案例；若要通过量子纠错来缓解错误，则需要更多量子比特（预计需要 2000 万个量子比特才能破解标准 RSA 加密 [12]）。

近年来，混合经典/量子算法更受青睐 [13]。这些算法是参数化量子算法，会与经典优化器交互，以更新和优化其参数。这类算法的示例包括量子近似优化算法（QAOA） [2]以及变分量子本征求解器（VQE） [14]。使这些算法对含噪中等规模量子计算颇具吸引力的特性之一是，实现量子优势所需的量子比特数量可能只需数百至数千个 [15]。尽管当前量子计算芯片的发展水平尚未达到这一阶段，但预计不久的将来即可实现 [16]。

尽管这些算法看起来前景广阔，但在实际应用中也会遇到困难。其中最突出的是对 QAOA 的参数进行经典优化的难题，因为已有研究表明，这种优化是 NP 困难 [17]。此外，由于该算法同时涉及经典计算和量子计算，因此必须评估量子控制栈中的流水线，以确保计算效率。第五节将进一步详细讨论这项评估，并指出量子控制栈中的瓶颈。

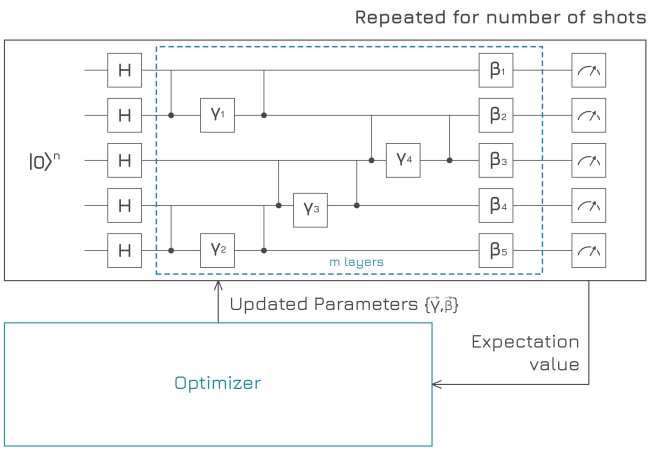

图 2. QAOA 布局的图示。经典计算机更新 QAOA 的参数 $\vec{\beta}$ 和 $\vec{\gamma}$。每次迭代后，利用测量统计数据评估问题实例的代价函数，随后由经典优化器更新 QAOA 参数。

A. 量子近似优化算法（*QAOA*）

QAOA 是参考文献 [2] 中提出的一种变分量子算法，旨在解决组合优化问题。QAOA 是一种分层算法。各层由一个“代价”酉算子和一个“混合”酉算子组成，每个酉算子都由一层参数化门实现。其基本思想是，代价酉算子对待求解的问题实例进行编码，而混合哈密顿量则生成基态的叠加态，以便随后找到产生最优问题解的状态。该优化属于有界约束优化，因为参数是取值范围限制在 $[0, 2\pi]$ 内的旋转角。图 2 展示了该算法一般结构的示意图。请注意，只有单量子比特门带有参数，而双量子比特门是静态的。本研究采用的实现使用 2 层，对于较小的问题规模，这被认为已足够 [18]。此外，在无噪声情形下，增加层数可以提高精度 [2], ，但在含噪量子计算机上未必如此。需要指出的是，虽然这里所有问题规模对应的层数保持不变，但每层的电路深度可以随问题规模增大。一般来说，可以在 2 个量子比特之间设置约束，这对应一组双量子比特门。在最坏情况下，每个量子比特都与其他所有量子比特共享约束，因此会产生 $^{n}{}_{(n-1)}{}_{2}$ 个约束，其中 n为量子比特数。在所用的基准测试集中，约束数与量子比特数的比率保持不变（每个量子比特对应 4 个约束），因此增长呈线性。

IV. 控制硬件基准测试

A. 基准测试指标

为了对量子控制硬件进行基准测试，必须评估多项指标，包括运行时性能和结果准确度，如前期工作中所述并实现的工作 [5]。本文着重关注运行时性能。对于能够在实验装置中访问量子芯片的工具用户而言，结果准确度也很重要。然而，在本研究中，我们使用一个子程序来模拟无噪声量子电路（如图 2 所示），以确保基准测试和性能分析具有可复现性，且不受低温恒温器和量子设备校准是否可用的影响。因此，尽管我们的工具也能评估结果准确度，但本文其余部分不再对此进行讨论。

本文介绍的基准测试工具主要关注以细粒度分辨率测量各种计算步骤的运行时间。测量的步骤包括：

- 编译：由主机 CPU 上的控制软件将量子电路编译为 Q1ASM 指令（Qblox 使用开源的多厂商 Quantifysoftware [9]）。
- 调度（执行）：将电路（量子门、测量）和量子比特重置作为脉冲，在（虚拟）量子芯片上执行。
- 通信：主机 CPU 与量子控制硬件之间的通信，包括可能需要多轮通信来设置模块并发送编译后的指令。通信类别进一步细分为：
  - 停止：停止调度的命令。作为一项安全预防措施，它既用于确保模块在接收新调度前已准备就绪，也用于在获取测量数据后确保模块就绪。– 准备：通过以太网连接将编译后的波形调度发送至控制模块和读出模块，随后这些模块会使时序器做好准备。– 启动：启动波形调度的命令。– 等待完成：主机 CPU 持续等待并检查调度是否已完成，此处不计调度执行时间（在我们进行的性能分析测量中，此运行时间包含调度执行时间，但会减去后者，以单独计算等待完成开销）。– 获取采集数据：将测量结果从读出模块传回主机 CPU。
- 总计：量子-经典混合算法的总运行时间。

当前的实现具有多项优势。首先，由于主机 CPU 与量子控制硬件直接连接，因此不存在 API 调用或长距离高延迟连接等因素。其次，可以精细控制对量子运行时间的测量。实际上，可以直接访问所有已调度指令的流水线，从而精确测量运行时间。此外，还可以深入了解主要计算步骤中的不同子过程。

B. 方法

本工作中用于基准测试和性能分析的初始化配置采用一台主机 CPU（Intel Core i5 第 10 代 i5-10210U、8GB 随机存取存储器、Nvidia GeForce MX250），并将其连接至 Qblox

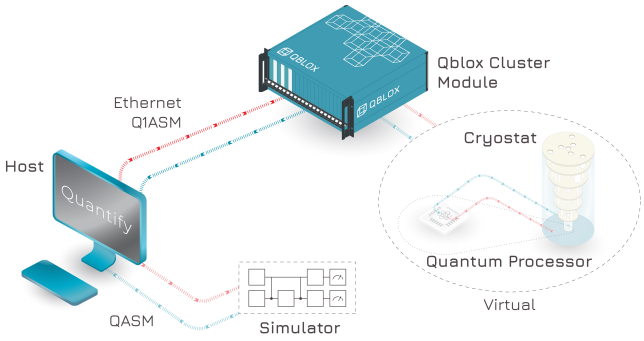

图 3. 用于对控制硬件进行基准测试的初始化方式。主机 CPU 通过以太网向 Qblox Cluster 发送 Q1ASM指令；该设备执行操作时的表现，就如同连接着真实的传输子量子比特。Cluster 为虚拟芯片报告的随机测量结果，会在主机 CPU 中替换为使用量子汇编语言模拟 QAOA 电路（例如图 2 所示电路）得到的测量结果。

Cluster，连接方式为 1 Gbps 以太网连接（见图 3）。对于 14 个量子比特，本工作中的 Cluster 使用三个量子控制模块和三个读出模块，为单量子比特门和测量生成脉冲。由于双量子比特门是静态的，因此其调度始终相同，我们实际上不会在单独的控制模块中生成相应的脉冲（但会在报告的调度时间中计入其量子门时间）。每个模块都采用一个名为 Q1处理器的时序确定性、图灵完备处理器来执行控制和读出任务。Q1处理器运行的 Q1ASM 汇编代码由主机 CPU 上的 Quantify 编译。为忠实重现实验条件并估算运行时间，Cluster 执行操作时会假定自己连接着实际的传输子量子比特。这意味着 Cluster 会生成脉冲，并测量其输入端口处的信号。在此场景下，由于这些端口未连接，Cluster 实际测量的是噪声，并在检索采集步骤中传回随机测量结果。这些测量结果会返回主机 CPU，但随即被丢弃，并替换为模拟电路的测量结果，随后由优化器用于参数更新。

电路模拟通过 Qiskit Aer模拟器 [19], 实现，但今后也可以集成任何支持量子汇编语言的模拟器。请注意，运行 QAOA 这类混合量子算法需要根据测量结果进行反馈，因此运行基准测试需要进行模拟（当然，如果有条件，也可以使用实际的量子设备）。表 I 报告了所考虑的量子门和操作的设备参数，这些参数参考了文献 [20]。此外，为简化编译，我们假设采用全连接拓扑（关于使用 S WAP门所需开销的估算，参见第 VI 节）。

QAOA算法使用 QPack套件 [5]实现。具体而言，我们针对 4 到 14 个量子比特的问题规模，研究用于最大割问题的量子近似优化算法。对于每个量子比特数，我们

表 I 所考虑的量子操作执行时间。

被动复位	主动复位	单量子比特门	双量子比特门	测量
200 μ秒	1 μs	40 纳秒	100 纳秒	500 纳秒

图 4 所示的 4 个量子比特量子近似优化算法在 1000 次采样下每次优化迭代的平均运行时间，见表 II。

计算步骤	测得的运行时间 [毫秒]
量子电路执行	11.7
重置（被动）	200
重置（主动）	1
调度执行（被动）	211.7
调度执行（主动）	12.7
启动	57.11
准备（基线）	207.2
准备（并行）	73.9
等待完成	56.3
获取采集数据	58.9
Stop	19.5
通信总计（基线）	388.1
通信总计（并行）	254.8
编译	84.2
优化器	0.47
总计（基线）	694.8
总计（主动复位）	495.7
总计（主动复位 + 并行）	348.8

考虑一个单一问题实例，并从相同的初始参数出发运行基准测试 40 次。由于（模拟的）测量结果具有概率性，每次达到收敛所需的优化迭代次数各不相同。为将这一点纳入考量，第 V 节中的时间结果按迭代次数和运行次数取平均值。所使用的经典优化器是 SHGO 优化器（单纯形同调全局优化） [21]，该优化器包含在*scipy.optimize* 库 [22] 中。我们注意到，对于 4 个量子比特，每次迭代耗时 0.47 ms。每次迭代使用 1000 次采样，因为事实证明，对于所研究的小型问题规模而言，这一采样次数能够提供足够准确的优化结果 [5]。QAOA 调度使用单量子比特门和双量子比特门，量子门总数最多为 64 个，并在末尾进行测量，因此电路时间总计最多为 14.8 μ秒。然而，编译后的调度通常会有更短的执行时间，因为优化器将参数设为 0 的量子门会被编译器取消或合并。

V. 结果

A. 4 个量子比特基准测试与已处理的瓶颈

量子电路执行（即量子门和测量，不包括量子比特重置）通常构成量子算法或实验的核心。然而，图 4 中的性能分析结果（另见表 II）显示，计算的主要部分并非 耗费在量子电路执行上。事实上，可以计算得出，基线总运行时间是量子电路执行运行时间的 59.4 倍（694.8 毫秒对比 11.7 毫秒）。因此，减少这巨大的开销将对实际应用产生重大影响。基线计算中最耗时的步骤是

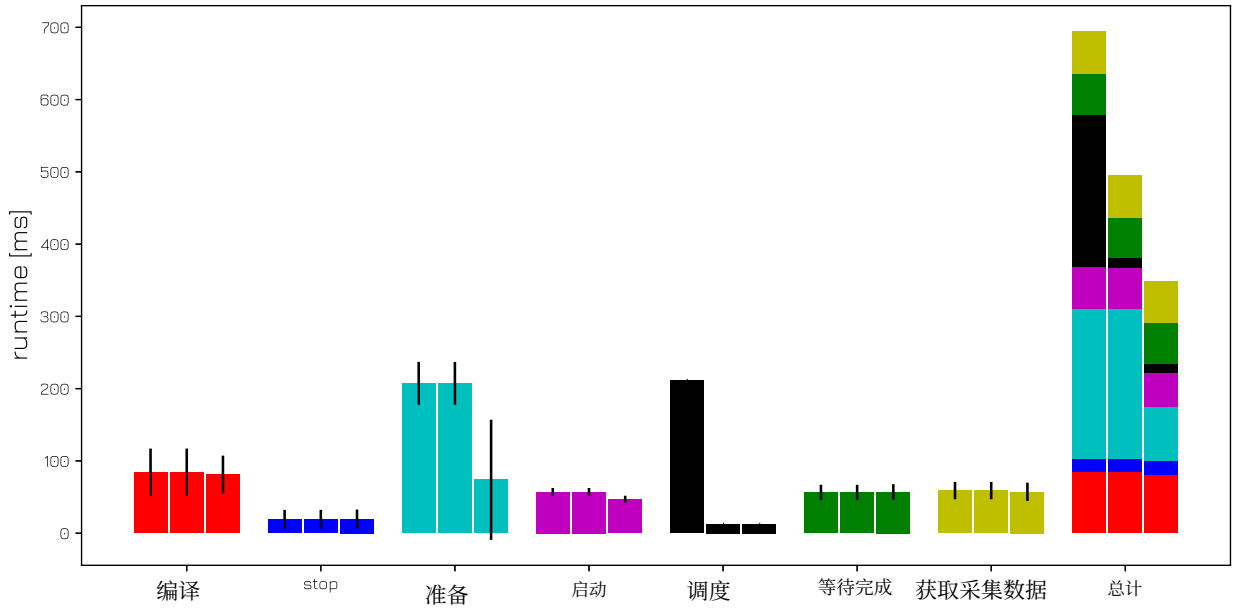

图 4. 4 个量子比特上的量子近似优化算法每次优化迭代的平均运行时间，每组数据均测量了 1000 次采样（具体数值见表 II）。具体而言，结果是对优化迭代次数以及基准测试的 40 次运行取平均值。误差线表示标准差（均方根偏差）。图中展示了基线（左列）、采用主动复位（中列），以及采用主动复位并结合已实现的并行模块准备时的运行时间。请注意，并行模块准备与采用被动还是主动复位无关。

按耗时从高到低依次为：被动量子比特重置（通常如此）、主机 CPU 与量子控制硬件之间的通信（具体而言是模块准备步骤），以及调度编译。我们注意到，每次运行的运行时间标准差可能很大——例如，我们测得最长的一次运行耗时 24.49 秒，共进行了 34 次迭代；最短的一次耗时 5.62 秒，共进行了 8 次迭代。尤其对于大型或含噪系统，经典优化器需要进行大量迭代。因此，考虑每次迭代的平均运行时间最为相关。

我们首先讨论如何缓解被动量子比特重置造成的瓶颈。可以通过实现主动量子比特重置来实现这一点 [23]。我们假设主动复位所需时间约等于量子比特测量时间加上校正门所需时间，并对总时间采用较为保守的估计，即 $1\ \mu$ s。撰写本文时，与 Cluster 的反馈周期已缩短至 320 ns，因此复位周期可达到 500–700 ns（具体取决于飞行时间和读出脉冲持续时间）。据估计，这使调度执行速度提高了

$$\frac{211.7}{211.7-199} = 16.67 \text{ times}, \tag{1}$$

从而整体加速比达到

$$\frac{694.8}{694.8-199} = 1.40 \text{ times}. \tag{2}$$

与基线的比较如图 4 activereset（中间列）所示。

出现的第二个执行瓶颈是通信开销（388.1 ms）。这远高于通过一根短以太网电缆测得的 2.5–3 ms 裸延迟。我们观察到，这一开销主要源于对 Cluster 中控制模块和读出模块的初始化（“准备”）。控制软件（此处为 Quantify）会依次准备这些模块。我们通过多线程实现了这一步准备过程的并行化。具体而言，编译后的波形不再按模块依次加载，而是并行加载。

如果并行化能够完全实现，并行化准备的运行时间将为

$$388.1 \text{ ms} - \frac{(388.1 \text{ ms} - 2.5 \text{ ms}) \times 4}{5} = 79.6 \text{ ms}. \tag{3}$$

然而，并未达到这一最大加速比，原因是存在无法并行化的开销。

实际提升效果如图 4 所示：主动复位+ 并行模块初始化（右侧列）。采用此实现方式测得的加速比为：

$$\frac{694.8}{694.8-(388.1-254.8)} = 1.24 \text{ times}, \tag{4}$$

而如果将其实现于主动量子比特重置之上（此时开销瓶颈更加明显），加速比则为

$$\frac{694.8-199}{694.8-199-(388.1-254.8)} = 1.37 \text{ times}. \tag{5}$$

请注意，图 4 中的并行准备步骤显示出相对较大的标准差。我们推测这是由

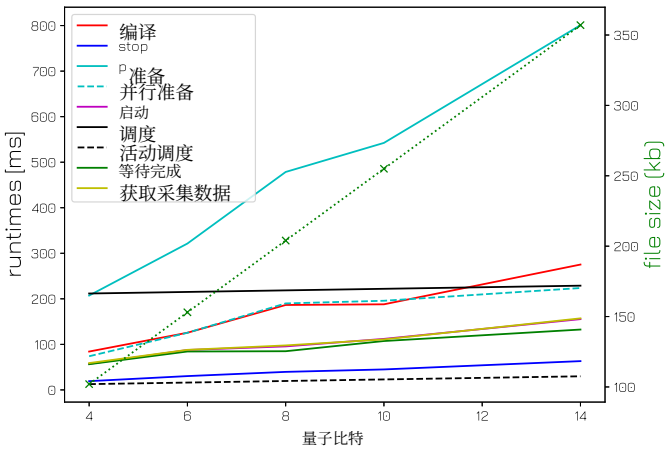

图 5。**左侧纵轴**：量子近似优化算法在 4、6、8、10、14 个量子比特上的每次优化迭代平均运行时间，每种情况的测量次数均为 1000 次。具体而言，结果是对优化迭代次数以及 40 次基准测试运行取平均得到的。**右侧纵轴**：调度数据大小，见带叉号标记的虚线。

通信期间的冲突造成的，因为采用多线程时可能会发生这种情况。

B. 从 *4* 个量子比特扩展到 *14* 个量子比特

我们考察运行时间随量子比特数量的增长情况，以深入了解更大规模系统中的瓶颈。图 5 展示了最多扩展到 14 个量子比特的性能分析结果，表明硬件准备步骤的增长斜率最大，并且主导了执行时间。与基线准备步骤相比，采用并行准备步骤在 14 个量子比特时实现了 3.58 倍的加速。即使采用并行准备，执行时间仍呈线性增长，不过增长幅度较小。我们认为，这种线性增长源于准备命令期间发送的调度数据量增加，以及模块就绪所需时间增加。对数据大小的测量表明，数据量随量子比特数量线性增长，每个量子比特约增加 25 kb，这与准备时间的线性增加相吻合。

C. 外推至 *50*+量子比特

由于变分量子算法可能具有实用价值的规模约为 400 个量子比特 [15], ，因此将当前结果外推至这一规模会很有意义。然而，我们不能简单地对 4 个量子比特到 14 个量子比特的测量数据进行线性拟合，然后将其延伸到如此大的规模，因为增长阶数（线性、二次或其他形式）取决于许多因素和假设。我们将讨论在方形网格连接性下，对量子近似优化算法优化器、调度、准备步骤、编译以及 SWAP门数量的预期。

对于优化器而言，迭代次数很大程度上取决于代价函数的光滑性以及局部极小值的数量。这些因素会增加优化器找到全局最优解的难度，从而影响算法的准确率指标。由于该函数是对量子算法的黑盒评估，因此这将很大程度上取决于问题实例和量子设备噪声。不过，关于优化所需时间，可以给出两点说明。具体而言，对于 SHGO算法，已知 SGHO 很难在变量数超过 10 时取得良好结果 [21]。由于 QAOA 的优化在问题规模还很小时就已达到这一程度，因此 SHGO 的性能可能并不适合大规模 QAOA。当前研究正在探索利用机器学习解决这一优化问题，但结果表明，训练这些算法非常耗时，即使对于小规模问题也需要数小时 [24], [25]。关于 QAOA 优化时间复杂度的第二个问题是参数数量的增长。这取决于 (a) 约束数量和 (b) QAOA 层数。请注意，参数总数是这两个因素的乘积（再乘以一个较小的常数）。目前，层数随问题规模增长的情况仅通过经验确定。然而，约束数量最多呈二次增长（$\binom{n(n-1)}{2}$ 其中 n 为量子比特数），适用于全连接图；在假设所有节点都相连的情况下，约束数量最少呈线性增长。

上述讨论的 QAOA 参数潜在的二次缩放会直接反映在调度执行时间上。调度文件的大小也会相应增长，从而增加（并行）准备步骤期间加载编译后的调度所需的时间。对于最多 14 个量子比特的情况，我们观察到，模型波形的描述和调度本身（即对波形进行排序并指定参数以形成调度）根据模块类型不同，占据数据的不同比例。对于读出模块，数据由 $\sim 87.5\%$ 的波形描述和 $\sim 12.5\%$ 的调度指令（测量）组成。由于每个量子比特只有一次最终测量，因此对于更大的量子电路，这些比例不会发生变化。对于控制模块，数据由 $\sim 12.5\%$ 的波形描述组成；由于扩展电路规模时没有引入新的量子门类型，该比例保持不变。调度描述占数据的 $\sim 87.5\%$，对于更大的系统，这一部分可能增长，但对于该基准测试而言几乎没有增加（我们仅观察到，每个量子比特的调度描述在每个指令文件中的变化最多为 1 KB）。我们将其归因于以下事实：对于所使用 QPack 基准测试中选定的问题实例 [5], ，如第 III-A 节所述，约束与量子比特的比例呈线性关系，因此平均而言，每个量子比特的调度不会增加（不计 SWAP门开销，见下文）。在这一规模下，由于调度长度增加导致的文件大小增长可以忽略不计，因此唯一的驱动因素是总数据量的线性增长，这是由编译文件数量的线性增加造成的（每个量子比特对应一个控制文件和一个读出文件）。然而，如果问题采用不同的结构，每个量子比特对应的约束数量可能会随问题规模增长。此时，由于最坏情况下只有控制调度本身可能呈二次增长，我们预测，对于相对较少的量子比特，数据大小以及准备步骤运行时间仍将呈线性增长，尽管后者的具体数值难以确定。预计影响最大的一项因素是

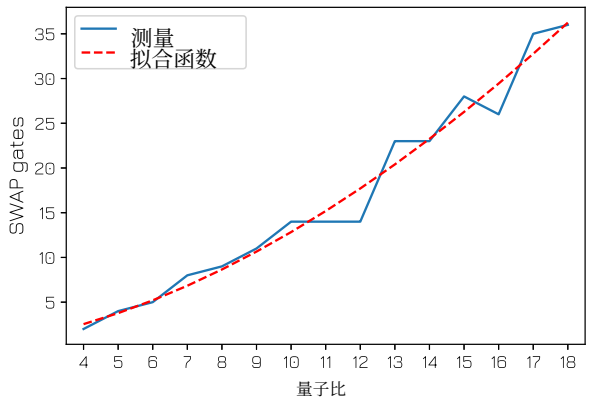

图 6. 方格量子比特布局下，QAOA算法所需 SWAP门数量的测量结果。对结果进行多项式函数拟合，得到指数为 1.7。

调度加载到时序器上的过程，这会随问题规模增大而延长。连接的带宽可能会在某个时候形成瓶颈；不过，在每个量子比特约 25 KB 的文件大小下，以太网连接（1 Gbps）的带宽不会对近期量子计算机造成问题（例如，传输 14 个量子比特所需的调度大约需要 360 μ 秒，另加少量延迟开销）。

编译的扩展规模更难以预测。调度的二次增长会使编译步骤的某些部分呈现二次扩展，但目前尚不清楚其影响有多大。此外，给出的编译运行时间针对全连接，因此不包括在连接受限时编译 SWAP门所产生的开销（见下文）。在 Qiskit 中，这部分开销难以单独分离，也难以与 Quantify 中不同的调度格式结合。

为简便起见，我们假设采用全连接，但传输子量子比特通常不具备这种连接，因此每个双量子比特门都必须经过编译，并使用若干个 SWAP门。这取决于问题实例、约束数量和量子比特布局，通常会导致调度时间增加。为估算这部分开销的增长情况，我们使用 Qiskit 的调度转译器[26]，针对相同的 QAOA算法编译方格布局所需的 SWAP门。选择这种布局是为了模拟参考文献 [20] 中的芯片以及 Google Sycamore处理器 [1]。结果如图 6 所示。我们使用多项式函数拟合这些结果，得到幂指数 1.7。SWAP门数量的增长情况是合理的：其上限为三次增长（双量子比特门数量呈二次增长，每个双量子比特门所需的 SWAP门数量最多呈线性增长），下限为 0（问题恰好映射到量子比特布局上的情况）。

如本节所述，由于 QAOA 层数、调度规模和

表 III 主动复位和并行模块准备情况下，每次优化迭代的实测平均运行时间（按降序排列），并线性外推至 50 个量子比特，每次各进行 1000 次测量。

执行步骤	运行时间 [秒]
编译	0.923
并行准备	0.754
启动	0.479
获取采集数据	0.478
等待完成	0.388
Stop	0.213
调度（主动复位）	0.090
总计	3.325

编译的扩展规律尚不明确，因此对测得的平均运行时间进行外推存在不确定性。尽管如此，我们仍采用最乐观情形下的线性外推，将结果推算至 50 个量子比特，以粗略估算所需的运行时间（见表 III）。只有调度时间包含了额外 SWAP 门数量的多项式外推。这些数字表明，下一个主要瓶颈是编译（事实上，在图 5 中，量子比特数为 14 时就已如此）。如前所述，实际耗时预计会远高于线性估算值。我们在第 VI 节提出了缓解这一编译瓶颈的方法。

VI. 讨论

虽然本文给出的运行时间测量结果将运行时间分解为几个主要类别，但 Q-Profile 还能进一步将其细分为更精细的模块。随着新的瓶颈不断出现，有必要对计算步骤进行更深入的性能分析。正如第 V-C 节所讨论的，预计其中一个瓶颈是量子电路编译。目前，每次优化迭代都需要从头执行这一步骤。然而，量子电路仅包含参数不同的相同量子门。一种解决方案是将量子电路调度参数化：仅将更新后的参数上传至控制硬件，而将通用电路结构保存在控制硬件上。这将大幅缩短量子电路编译时间，并进一步加速变分量子算法。此外，还可以考虑采用专用硬件加速。需要注意的是，如果对调度进行优化，使其能够使用并行量子门来最佳匹配量子比特布局，那么所需的编译时间将比本文所述的更长。量子电路优化本身就是一项具有挑战性的任务，其中包括针对特定情境改进电路，例如用于量子近似优化算法的电路 [27]。将这项工作纳入编译过程会进一步延长编译时间，因此需要权衡编译时间与电路优化。

我们的工具与 CLOPS 基准测试 [10] 有着共同的目标。该基准测试旨在衡量的不仅是量子芯片上执行量子门的速度，还包括经典计算对指令进行调度和编译，以及收集结果并执行后处理的速度。IBM 的结果 [10] 表明，量子控制硬件存在瓶颈，而且设备越大，瓶颈越明显，因为所需的控制硬件更多，

同时初始化和加载所需的时间也会增加。具体而言，正如预期，IBM 的云端设置所产生的大部分开销来自通信，其程度甚至远高于我们的结果。作为一种算法，CLOPS 基准测试采用参数化量子体积（随机）电路，以体现量子计算机的典型用法。其得分受实验轮次数、模板、参数更新和量子体积层数的影响。虽然我们的工具也可用于运行相同的电路，但本文选择将其用于量子近似优化算法（QAOA），因为我们认为，对广受关注的特定算法进行性能分析同样切合实际且恰逢其时。

VII. 结论

我们提出并展示了 Q-Profile，这是一款精准且高效的工具，可用于对量子控制硬件和软件进行基准测试与性能分析。随着人们希望利用量子计算实现加速，对量子系统进行性能分析变得日益重要。我们将 Q-Profile 应用于 QAOA，以反映近期的实际应用场景；此外，它还可扩展至其他量子算法，并支持更多量子比特，以研究开销及其特征如何随系统规模变化。该工具揭示了当前（变分）量子计算中的各种瓶颈，并展示了未来对量子控制堆栈进行性能分析的能力。我们还改进了硬件控制软件，以展示性能分析的有效性。通过这项工作，我们证明了 Q-Profile 的实用价值，并为硬件和软件开发人员提供了有关如何发现系统性能瓶颈的见解。性能分析功能已集成到开源软件 Quantify [9] 中。

参考文献

[1] F. Arute, K. Arya, R. Babbush 等，“使用可编程超导处理器实现量子霸权”，《自然》，第 574 卷，第 7779 期，第 505–510 页，2019 年 10 月。 [在线]。网址：http: //dx.doi.org/ 10 1038 41586 019 1666 5 。/s - - - E. Farhi、J. Goldstone 和 S. Gutmann，“一种量子近似优化算法”，2014年。 [在线]。网址：https://arxiv.org/abs/ 1411.4028[3] D. A. Fedorov、B. Peng、N. Govind 和 Y. Alexeev，“变分量子本征求解器方法：简要综述与最新进展”，2021年。 [在线]。网址：https://arxiv.org/abs/ 2103 08505 。P. V. Klimov、J. Kelly、J. M. Martinis 和 H. Neven，“用于学习量子处理器控制参数的蛇形优化器”，2020年。 [ ] 2006 04594 在线。网址：https://arxiv.org/abs/。K. Mesman、Z. Al-Ars 和 M. Moller，“QPack：将量子近似优化算法作为量子计算机的通用基准测试”，2021 [ ] 2103 17193 。在线。网址：https://arxiv.org/abs/。P.-L. Dallaire-Demers、M. Ste.chły、J. F. Gonthier、N. T. Bashige、J. Romero 和 Y. Cao，“费米子量子模拟的应用基准测试”，2020年。 [在线]。网址：https://arxiv.org/abs/ 2003.01862[7] S. Martiel、T. Ayral 和 C. Allouche，“以应用为中心、与硬件无关且可扩展的量子协处理器基准测试”，第 2 卷，2021年，第 1–11 页。 [在线]。网址：https://doi.org/10. 1109 2021 3090207 /TQE. 。T. Lubinski、S. Johri、P. Varosy、J. Coleman、L. Zhao、J. Necaise、C. H. Baldwin、K. Mayer 和 T. Proctor，“面向量子计算的应用性能基准测试”，2021年。 [在线]。网址：https://arxiv.org/abs/2110.03137

[9] Quantify 软件包可在 https://gitlab.com/quantify-os 获取；Q-Profile 则作为 Quantify-Scheduler 的一部分，位于 https://gitlab.com/quantify-os/quantify-scheduler/-/blob/v0.12.0/quantify scheduler/ gettables profiled.py。[10] A. Wack、H. Paik、A. Javadi-Abhari、P. Jurcevic、I. Faro、J. M. Gambetta 和 B. R. Johnson，“质量、速度与规模：衡量近期量子计算机性能的三个关键属性”，2021 [ ] 2110 14108 。在线。网址：https://arxiv.org/abs/。N. P. de Leon、K. M. Itoh、D. Kim 等，“量子计算硬件的材料挑战与机遇”，《科学》，第 372 卷，第 6539 期，文章编号 eabb2823，2021年。 [在线]。网址：https: //www.science.org/doi/abs/ 10 1126 2823 。/science.abb C. Gidney 和 M. Ekera，“如何使用 2000 万个含噪量子比特在 8 小时内分解 RSA 整数”，《量子》，第 5 卷，第 433 页，4 月，2021 [ ] 10 22331 2021 04 15 433 。在线。网址：https://doi.org/。/q- - - - M. Cerezo、A. Arrasmith、R. Babbush、S. C. Benjamin、S. Endo、K. Fujii、J. R. McClean、K. Mitarai、X. Yuan、L. Cincio 等，“变分量子算法”，《自然·物理评论》，第 3 卷，第 9 期，第 625–644 页，2021年。 [在线]。网址：https://www.nature.com/articles/ s42254-021-00348-9[14] A. Peruzzo、J. McClean、P. Shadbolt 等，“光子量子处理器上的变分本征值求解器”，《自然·通讯》，第 5 卷，第 1 期，第 1–7 页，2014年。 [在线]。网址：https://www.nature.com/articles/ncomms5213[15] G. G. Guerreschi 和 A. Y. Matsuura，“最大割的量子近似优化算法需要数百个量子比特才能实现量子加速”，《科学报告》，第 9 卷，第 1 期，2019年 5 月。 [在线]。网址：https: //doi.org/ 10 1038 2 41598 019 43176 9 。% Fs - - - J. Gambetta，“IBM 扩大量子技术规模的路线图”，2020年。 [在线]。网址：https://research.ibm.com/blog/ibm-quantum-roadmap [17] L. Bittel 和 M. Kliesch，“训练变分量子算法是 NP 困难的”，《物理评论快报》，第 127 卷，第 12 期，2021年 9 月。 [ ] 10 1103 2 127 120502 在线。网址：https://doi.org/。% Fphysrevlett. 。M. Y. Niu、S. Lu 和 I. L. Chuang，“优化量子近似优化算法：成功概率和运行时间对电路深度的依赖关系”，2019年。 [在线]。网址：https://arxiv.org/abs/ 1905 12134 。“IBM Aer Quasm 模拟器”，https://qiskit.org/documentation/stubs/qiskit.providers.aer.QasmSimulator.html。[20] S. Krinner、N. Lacroix、A. Remm 等，“在距离三表面码中实现重复量子纠错”，《自然》，第 605 卷，第 7911 期，第 669–674 页，2022年 5 月。 [在线]。网址：https://doi.org/ 10 1038 2 41586 022 04566 8 。% Fs - - - S. C. Endres、C. Sandrock 和 W. W. Focke，“用于利普希茨优化的单纯同调算法”，《全球优化杂志》，第 72 卷，第 2 期，第 181–217 页，2018年。 [在线]。网址：https://doi.org/10.1007/s10898-018-0645-y [22] P. Virtanen、R. Gommers、T. E. Oliphant 等，“SciPy 1.0：科学计算的基础算法”，《自然·方法》，第 17 卷，第 261–272 页，2020年。 [在线]。网址：https: //www.nature.com/articles/s41592-019-0686-2[23] D. Rist`e、C. C. Bultink、K. W. Lehnert 和 L. DiCarlo，“利用高保真投影测量对固态量子比特进行反馈控制”，《物理评论快报》，第 109 卷，文章编号 240502，2012 年 12 月。 [在线]。网址：https://link.aps.org/doi/ 10 1103 109 240502 。/PhysRevLett. 。S. Khairy、R. Shaydulin、L. Cincio、Y. Alexeev 和 P. Balaprakash，“用于量子近似优化的强化学习”，载于超算大会，第 19 卷，2019年，第 1–10 页。[25] M. Alam、A. Ash-Saki 和 S. Ghosh，“使用机器学习加速量子近似优化算法”，载于 2020 年欧洲设计、自动化与测试会议暨展览会（DATE），2020年，第 686–689 页。 [在线]。网址：https://ieeexplore.ieee.org/document/9116348 [26] “Qiskit Terra 转译器”，https://qiskit.org/documentation/apidoc/ transpiler.html。[27]M. Alam、A. Ash-Saki 和 S. Ghosh，“量子近似优化算法的电路编译方法”，载于 2020 年第 53 届 IEEE/ACM 国际微体系结构研讨会（MICRO），2020年，第 215–228 页。 [在线]。网址：https://ieeexplore.ieee.org/document/9251960