Clo基于张量网络的量子动力学按需仿真 s

Kemal Bidzhiev, * Aleksander Wennersteen, † Mourad Beji、Mario Dagrada、Mauro D' Arcangelo、Sebastián Grijalva、Anne-Claire Le Hénaff、Anton Quelle 和 Alvin Sashala Naik*PASQAL*，法国马西市莱昂纳尔·德·芬奇街7号，邮编*91300*（日期：2023年2月13日）

我们介绍了一种基于张量网络的仿真器，用于模拟可编程模拟量子处理单元（量子处理器）。该软件包完全集成于云平台，提供统一接口，既可在高性能计算集群上执行作业，也可将作业分发至量子处理设备。我们还介绍了中性原子量子处理器领域中的典型仿真用例，例如评估态制备脉冲序列的质量，以及通过对一组输入脉冲参数值进行并行扫描，求解由大量量子比特构成的系统中的最大独立集问题。

arXiv:2302.05253v1 [quant-ph] 10 Feb 2023

I. 引言

量子技术旨在利用量子力学定律，在一种能够比经典替代方案更高效地解决复杂计算问题的框架中处理信息。这一理念催生了量子信息科学的新时代，近年来，构建首批高性能量子处理单元（QPU）已成为一项重要目标。随着 QPU 的效能和可靠性不断提升，研究人员已通过实验展示了能够解决特定计算任务的方案，这些方案可能优于经典方法，例如采样问题 [1–4]，和量子模拟实现 [5, 6]。人们已基于多种不同的物理架构提出了 QPU 平台，包括中性原子、超导电路、囚禁离子和光子。

除了持续推进 QPU 的实验发展，模拟特定物理 QPU 架构的行为也日益重要，这一过程称为仿真。模拟器会对真实的物理约束进行编码，包括硬件特定的时间动力学和噪声效应。此外，它们还能够帮助预测 QPU 的预期性能并进行基准测试 [7]。用户因此可以先在高性能计算（HPC）后端执行测试作业，从而避免不必要的开销，然后再将完整作业提交至 QPU。此外，通过云访问集成 QPU 和模拟器后端，并提供选择

为用户提供后端，使其能够将高效工作流纳入全栈量子计算机。混合工作流会将合适的例程发送至量子处理器，同时在仿真器后端执行规模更大的计算任务。支持此类混合工作流和仿真量子设备的系统，已经为研究人员提供了宝贵的服务 [8–12]。

在大多数情况下，量子系统的模拟复杂度会随系统规模呈指数增长。应对这一挑战的一项重要工具是张量网络 [13]，，它提供了一种强大的结构，可高效地表示复杂系统。张量网络框架使量子系统得以高效模拟和研究；其中，与我们的目的最为相关的是，对以下对象进行数值研究：

有能隙局域哈密顿量的基态量子态的时间演化。此外，张量网络还被用于识别新的物质相 [14, 15]、作为数据处理的结构 [16, 17]，，或对具有大量量子比特的先进量子处理器实验进行数值基准测试 [5, 18]。

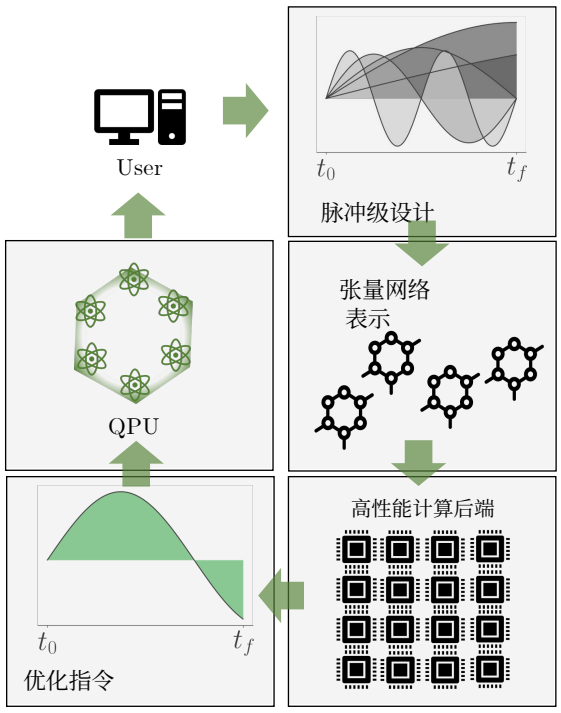


图 1. 仿真在大型量子系统工作流中的作用。脉冲级设计工具与相应量子态的张量网络表示相辅相成。执行时间演化的算法会同时考虑哈密顿动力学和现实的硬件条件，这些条件在高性能计算后端上进行模拟。利用这些结果，可以在量子处理器上准备优化指令。我们提出了一个按需将这些任务集成于一体的基于云的平台。

本文结构如下：在第 II 节中，我们回顾量子动力学数值模拟的基本设置，并介绍相关的张量网络方法。随后，在第 III 节中，我们介绍将 HPC 支持的数值算法集成到云中的框架。我们在 IV 节展示模拟器的应用，通过三个示例模拟中性原子量子处理器：生成 1D $\mathbb{Z}_n$有序态、制备 2D 反铁磁态，以及并行扫描脉冲参数以

* kemal.bidzhiev@pasqal.com
† aleksander.wennersteen@pasqal.com

在特定图上搜索最大独立集。最后，我们对系统的扩展进行评论，并讨论异构计算中的模拟环境。

II. 量子动力学的数值模拟

A. 量子系统的时间演化

量子比特，即二能级量子系统，是我们将要研究的量子系统的基本构成单元。一个 N 量子比特系统的量子态是 2^N 维希尔伯特空间中的一个元素。它可以表示为一个大小为 2^N 的复向量 $|\psi\rangle = \sum_x c_x|x\rangle$，其中我们选取 $\{|x\rangle\}$ 作为该空间的一组标准正交基。我们关注由每个格点上 $\hat{}\sigma^z$ 算符的本征态构成的标准正交基，称为计算基。其元素可写作 $|s_1 s_2 \cdots s_N\rangle$，其中每个 $s_i \in \{0,1\}$。字符串 $s_1 s_2 \cdots s_N$ 通常称为位串。因此，一般的量子态可写为：

$$|\psi\rangle = \sum_{s_1 \cdots s_N} c_{s_1 \cdots s_N} |s_1 \cdots s_N\rangle, \tag{1}$$

其中，$c_{s_1 \cdots s_N}$ 是一个复数，对应于一组给定索引 $s_1 s_2 \cdots s_N$所表示的测量结果的概率振幅。量子处理器接收一个初态，并通过一系列量子门和/或模拟脉冲使其发生演化。为了获得最终态，它会针对某个特定的测量基对其进行投影测量，并重复这一循环，直到满足某个条件。在这些操作中，计算量最大的是求出最终的时间演化态，下面将对此进行描述。

量子态的时间演化由薛定谔方程支配

$$i\frac{d|\psi\rangle}{dt} = \hat{H}(t)|\psi\rangle, \tag{2}$$

其中，我们考虑一个特定的随时间变化的哈密-ˆ尔顿 算符 $H(t)$，它由 J 随时间变化的控制 项和 $N(N-1)/2$ 二体相互作用 项$V_{ij} = V(\mathbf{r}ij)$之和构成；这些项取决于每一对量子比特的相对位置$\mathbf{r}ij$ i，j位于 $\{1,\ldots,N\}$中：

$$\hat{H}(t) = \sum_{j=1}^{J} \hat{H}^{(j)}_{\text{ctrl}}(t) + \sum_{i>j} V_{ij}\hat{P}_i\hat{P}_j. \tag{3}$$

ˆ与 P_i 一个由泡利矩阵构造的局部算符[1] 和 ˆ$H^{(j)}$ ctrl(t) 第 j 个控制项，一个随时间变化的表示实验参数作用的函数，这些参数会影响系统，通常使用波形发生器构造，并通过激光场或微波场施加。

微分方程 (2) 的解可以通过多种方法 [19] 获得：可以用 $\mathcal{U}(t) \approx \prod U(\delta t)$ 近似时间演化算符，其中 $U(\delta t) = \exp(-iH\delta t)$，适用于较小的时间间隔 δt，因此 ˆ H 在 δt 期间可以视为与时间无关。然而，对于大型系统，计算算符的指数可能代价高昂 [20]。还可以将 $U(\delta t)$ 展开为若干较简单项的乘积，以达到所需的精度，具体取决于 H [21] 的形式。此外，也可以尝试数值求解完整的微分方程，例如使用龙格-库塔方法 [22]。另一类方法是在量子计算机上实现模拟时间演化的算法，例如利用特罗特乘积公式 [23–25]，将指数分解为演化模块，或使用量子机器学习工具 [26, 27]。还可以尝试使用机器学习工具学习最终态矢量 [28]，，或应用蒙特卡罗技术 [29–31]。

最后，一种非常成功的方法是采用张量网络技术，这也是我们选择作为仿真框架一部分来实现的方法。对于不熟悉张量网络的读者，我们将在下文对其进行简要回顾。例如， [13, 16, 32, 33]中可以找到深入而详细的介绍。

B. 使用张量网络表示 Quantum 态及其演化

由于纠缠熵会随时间增长，仿真非平衡相互作用系统的时间演化颇具挑战性 [34]。张量网络运算本质上是线性代数运算，所涉及矩阵的大小随所研究系统的纠缠程度而扩展。不同类型的张量网络表现出不同的扩展方式，这取决于物理系统的几何结构和维度 [35——38], 然而，对于所有这些方法而言，所涉及的矩阵最终都会变得大到难以处理。此外，现有的时间演化算法中，只有一部分能够高效地编码长程哈密顿量的动力学 [39, 40], ，而这些哈密顿量是模拟二体相互作用（3）所必需的，其中包含形式为 $V_{ij} \sim 1/|\mathbf{r}_{ij}|^{\alpha}$的项。

为了模拟量子多体系统（1），我们将波函数编码为矩阵乘积态（MPS）[13, 41]。矩阵乘积态能够高效地表示N阶张量 $c_{s_1 \cdots s_N}$，将其表示为 N 较小张量$A[\sigma]^{s_\sigma}_{i_{\sigma-1} i_\sigma}$的乘积：

$$|\psi\rangle = \sum_{\{s\}}\sum_{\{i\}} A[1]^{s_1}_{i_1} A[2]^{s_2}_{i_1 i_2} \cdots A[N]^{s_N}_{i_{N-1}} |s_1 \cdots s_N\rangle, \tag{4}$$

其中，每个张量指标 $i_\sigma \in i$ 的取值范围为 1 到 χ_σ。键维数 χ_σ 控制每个 $A[\sigma]$的大小，进而决定模拟的计算复杂度[34]。

最成功的算法之一，能够处理长程相互作用，同时保持足够

---

[1] 泡利矩阵为$\hat{\sigma}^x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$、$\hat{\sigma}^y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$、和$\hat{\sigma}^z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$。我们用矩阵的克罗内克积表示它们对ˆ量子系统第 i^{th} 个格点的作用：$P_i = \hat{}$ $\mathbb{I} \otimes \cdots \otimes P \otimes \cdots \mathbb{I}$ $P \in \{\mathbb{I}, \sigma^x\ \sigma^y\ \sigma^z\}$，其中^^、^^占据第-个位置。随后通过矩阵乘积构造诸如ˆ P_iˆP_j 这样的复合算符。

小的键维数，是含时变分原理（TDVP）[42]。TDVP 将时间演化限制在 MPS 流形上 $\mathcal{M}(\psi[A])$，并通过将薛定谔方程的右端投影到切空间 $T\mathcal{M}(\psi[A])$：

$$\frac{d|\psi[A]\rangle}{dt} = -i\hat{\Pi}_{T\mathcal{M}(\psi[A])}\hat{H}|\psi\rangle. \tag{5}$$

由此得到的方程（5）是非线性的，并且耦合矩阵乘积态中的所有自由度。求解（5）的方法在精度控制和收敛性方面各不相同。在此介绍的模拟器中，我们采用两位点含时变分原理来处理核心数值模拟。实现细节见附录 B。

III. 平台架构

本节将介绍平台的整体架构。我们将说明如何将高性能计算集群与云服务集成，以实现量子设备模拟。我们将该模拟器称为EMU-TN，其纳入了特定量子处理器的约束条件，如第IIA节所述，并将第IIB 节中的张量网络算法作为主要数值后端。下文将介绍模拟器的输入、预处理，以及返回输出前的后处理。最后，我们将讨论云基础设施，包括编排、调度和分发。

A. 动力学信息的编码

我们的平台接收编码后的控制参数抽象表示 $H_{\text{ctrl}}(t)$ 作为输入。随后，它执行所需的量子动力学演化，对最终态进行测量，并将读出数据返回给设计工具。

EMU-TN 包含一个 JSON 解析器，这对于统一发送到集群的信息至关重要。我们将式（3）中的哈密顿量作为初始表达式，其中包含一组 J 控制场，以及寄存器中量子比特的位置 $\{\mathbf{r}_{ij}\}$。每个控制场 $\hat{H}^{(j)}{}_{\text{ctrl}}(t)$作用于寄存器中量子比特的一个子集$I^{(j)}$，

$$\hat{H}^{(j)}_{\text{ctrl}}(t) = \Omega^{(j)}(t)\sum_{i\in I^{(j)}}\hat{P}_i^{(j)}. \tag{6}$$

例如，如果第 j个控制是沿 z轴、幅值为 $\Omega^{(j)}(t)$ 的全局场，那么 $\hat{P}^{(j)} = \hat{\sigma}_i^z$ 和 $\hat{H}^{(j)}{}_{\text{ctrl}} = \Omega^{(j)}(t)\sum_{i\in\{1\dots N\}}\hat{\sigma}_i^z$。要解析这些信息，设计工具必须将 $\Omega^{(j)}(t)$ 指定为数值数组、初始值和最终值加总持续时间，或生成函数。此外，还必须用索引 $I^{(j)}$指定局域性。

最后，需要说明初态制备以及对该态的测量。初态通常选择在实验上便于制备的状态，例如由比特串指定的乘积态。中性原子量子处理器等平台还允许选择量子比特的位置，从而确定它们之间的相互作用强度。更一般地，初态条件可以由设计工具生成的标签提供。测量协议可以包含进行测量的一个基（或一组基），以及为生成统计数据以估算可观测量而进行的实验重复次数。只要与模拟量子处理器设备兼容，就可以通过作业提交的运行次数以及所需基的标签来指定这些内容。

目前已有越来越多的软件包提供工具，用于针对不同类型的量子比特架构和量子处理任务 [43–48]设计和定制脉冲序列。这些工具已应用于量子算法设计、量子最优控制以及多体量子系统动力学性质的研究。其中绝大多数直接或间接地采用量子态的状态向量表示，并作为开源工具供个人使用。EMU-TN 可作为求解器的替代方案，从而扩展这些设计工具能够处理的任务类型。

B. 云基础设施

我们介绍云平台如何将作业分派至经典模拟和量子计算后端。

1. 编排与调度

该基础设施如图 2所示，分为两部分：一部分是托管于 OVHCloud 、通过 Kubernetes 进行编排的公共云服务；另一部分是托管于私有数据中心、运行 Slurm 调度器的高性能计算集群。下面我们将介绍二者如何集成，以提供量子计算即服务。

Kubernetes [49] 是一种以云计算为核心的容器编排系统，适用于公有云、私有云或混合云。它简化了在一组资源上部署和运行容器化应用，以及扩展这些资源的过程。另一方面，Slurm [50], 是为高性能计算工作负载而设计的。它提供支持优先级的调度、作业队列、拓扑感知作业分配、预留和回填策略。这些特性使得 Slurm 等软件常被选来管理高性能计算集群，而不是采用专为其他计算范式定制的新解决方案。Slurm 会提交一个通用 Bash 脚本来运行作业，该脚本也可以包含一系列容器化应用。

可以在 Kubernetes Pod 上运行相同的工作负载，既可以将其作为负责接收和执行作业的长期运行服务，也可以使用兼容 Kubernetes 的调度器进行调度。不过，这些方法的缺点是，将本地高性能计算集群与 Kubernetes 调度器集成起来颇具挑战。如果不在集群上调度作业，而只是在 Kubernetes Pod 中运行作业执行服务，就会限制集群资源的利用效率，最终也会影响高性能计算应用的性能。

随着云计算与高性能计算日益融合，Kubernetes 与 Slurm 之间的扩展

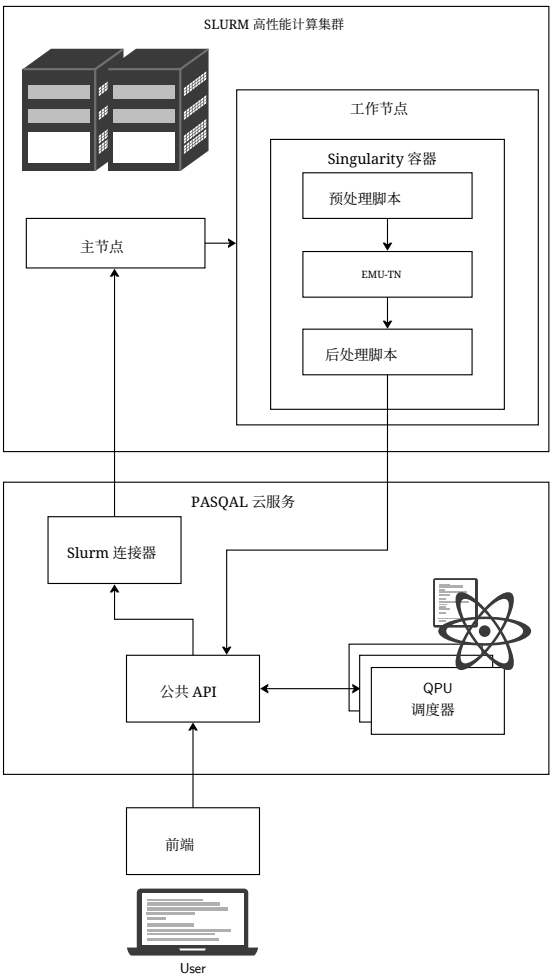


图2. 提供量子计算即服务的云平台架构。云服务部署在 Kubernetes 集群上，并采用微服务架构；每个微服务都向最终用户提供公共 REST API。运行`EMU-TN` 的高性能计算集群由英伟达的 10 台 DGX A100 系统组成（详情请参阅附录 A），并使用 Slurm（一种高性能计算集群管理和作业调度系统）进行资源管理和作业提交

近期取得了许多进展，例如参见 [51–53]。不过，这些方法通常侧重于将 Kubernetes Pod（即容器）提交到 Slurm 系统。这需要依赖 Kubernetes 来调度作业。Kubernetes 调度器并不原生支持在经常变化且高度异构的环境中运行，而这类环境包含多种量子处理器类型、中央处理器和图形处理器。在本研究中，我们为脉冲序列使用了自定义调度器。因此，我们没有采用与 [51–53] 中提出的方法类似的机制，而是选择了一项可直接提交常规 Slurm 作业的服务，因为它与现有硬件的集成效果更好，也更接近量子处理器上的作业调度方式。

此外，我们认为，在可预见的未来，量子-经典混合计算工作负载将构成相当大一部分量子作业。其中的经典计算部分通常可以通过 GPU 加速，这也促使人们需要能够在有需要时申请此类计算资源。例如，英伟达最近推出了一个用于混合量子-经典计算的平台，名为 Quantum 优化设备架构（QODA）[54]。

2. 作业派发

为了连接这两个集群，我们采用了最新引入的 Slurm REST API [55]，以便通过常规后端服务向 Slurm 提交作业。我们开发了一个“Slurm 连接器”服务，作为现有 PASQAL 云服务与高性能计算集群之间的桥梁。该服务会按需为作业创建控制脚本，并将其作为批处理脚本提交给 Slurm。该脚本负责执行作业的所有高层逻辑。该服务还负责为作业配置适当的资源，并插入后续与云服务通信所需的所有信息。

每个脉冲序列都以`JSON` 格式编码，并通过向作业提交端点发送 HTTP POST 请求，将其发送至 PASQAL 云服务。对于量子处理单元和模拟器，脉冲序列的格式是相同的。我们在 HTTP 请求的正文中添加了一个配置参数，使用户能够控制模拟器的参数，参见代码示例 1。这种设计可以在量子处理单元和模拟器上无缝执行相同的量子程序，同时还支持控制数值模拟参数，例如最大键维数和核心数。

作业提交端点的内部逻辑会验证脉冲序列和仿真配置。验证步骤会检查请求及其底层数据（例如序列和模拟器配置）的格式是否正确，请求的资源是否可供用户访问；对于真实设备，还会检查序列在物理上是否有效。随后，它会将请求分派给相应的调度器。在下面的代码示例 1中，我们展示了一个将序列化脉冲序列发送至 PASQAL 云服务以执行的示例。为构建这些序列，我们使用了`Pulser`[46], ，这是一个用于设计中性原子量子处理器序列的开源软件包。为与云服务通信，我们使用了 PASQAL Cloud Services Python SDK [56],，该 SDK 为最终用户提供了用于请求 API 的辅助函数。

代码示例 1. 通过云服务运行作业。

```
1 frompulser importPulse,Sequence, Register 2 frompulser.devices importMockDevice 3 fromsdkimport 配置, 端点, SDK 4 import json 5 6 #1. 定义量子比特寄存器 7 N = 6# 每边的量子比特数 8 reg = Register.square(side=N,spacing=7, prefix="q") 9 10 #2. 定义脉冲序列并编码为 JSON 11 seq = Sequence(reg, 模拟设备) 12 seq.declare_channel( 13 name=" ch0", 14 channel_id="rydberg_global" 15 ) 16 seq.add( 17 Pulse.ConstantPulse( 18 duration=1000, # 纳秒 19 amplitude=2, # 弧度/微秒
```

```
20 失谐=-6,# 弧度/微秒 21 相位=0), 22 通道="ch0" 23 ) 24
encoded_seq = seq.to_abstract_repr() # JSON 文件 25 26 # 3.
将编码后的序列发送至 PASQAL 云服务 27 # 在此获取 API 密钥
28 # https://portal.pasqal.cloud/api-keys 29 30 endpoints =
Endpoints( 31 core="https://apis.pasqal.cloud/core", 32
account="https://apis.pasqal.cloud/account" 33 ) 34 35 sdk =
SDK( 36 client_id="my_client_id", 37 client_secret="my_client_
secret", 38 endpoints=endpoints, 39 ) 40 41 # 配置作业 42 my_
job = {"runs": 1000}#采样次数 43 config = 配置(#模拟器配置
44 dt=10.0, 45 precision="normal", 46 extra_config={"
max-bond-dim": 100} 47 ) 48 49 # 在云服务上创建批次，等待执
行完成并获取结果 50 batch =sdk.create_batch( 51 encoded_
seq, 52 device_type="EMU_TN", 53 jobs=[my_job] ,54
configuration=config, 55 fetch_results=True, 56 ) 57 res = [
job.result for jobinbatch.jobs.values()]
```

在此背景下，EMU-TN 被构建为 Singularity 容器，并作为 Slurm 作业执行。Slurm 作业包含数值模拟，前后分别由定制脚本负责与云平台通信。由于脉冲序列此前已通过验证，预处理脚本只需将用户配置转换为 EMU-TN 所需的格式。模拟完成后，后处理脚本会收集结果，并与云服务通信，确保结果得到正确存储且可供前端访问。这种架构使模拟器能够独立运行，也能轻松地在整个平台之外执行。此外，若未来有此需求，也可以轻松扩展架构，以实现平台与模拟器之间更频繁的双向通信。

3. 其他考量

将二进制结果或经过高度压缩的张量状态存储在列式数据库中效果很好，因为它们可以序列化为较短的文本字符串。对于较大的数据集，这可能导致数据库性能下降，最终数据集会大到超出特定数据库实现的处理能力。因此，在这类情况下，对象存储可能会很有用，例如用于存储模拟器生成的张量网络表示。由于获取此类状态的计算成本很高，该平台会将其作为序列化对象提供，以便根据数值算法。

第二个考量是，并非每个用户请求都对应着可行的计算目标。某些脉冲序列（例如会引入陡峭变化的序列）通常需要更大的矩阵大小才能进行模拟。算法也可能无法达到所需的精度。候选脉冲序列和量子系统需要由用户从物理角度进行研究，并通过针对较小系统的精确对角化方法逐步验证；这一迭代过程不可避免。选定一个有前景的作业后，数据会在云核心服务中反序列化，验证脚本随后会检查：基于现有量子处理单元的模拟设备所对应的量子程序是否有效。这样可确保脉冲序列能够在硬件上实际实现。Slurm 连接器还会对 Slurm 配置进行一些简单验证。

IV. 应用

本节讨论 EMU-TN 框架的一些应用，这些应用的灵感来自中性原子量子处理器 [57–59] 可以执行的典型任务。原生哈密顿量将采用以下特殊形式（3）：

$$\hat{H}(t) = \frac{\Omega(t)}{2}\sum_{i=1}^{N}\hat{\sigma}_i^x - \delta(t)\sum_{i=1}^{N}\hat{n}_i + \sum_{i>j}\frac{C}{|\mathbf{r}_{ij}|^6}\hat{n}_i\hat{n}_j \quad (7)$$

其中，$\hat{n} = |1\rangle\langle 1| = (\mathbb{I} - \hat{\sigma}^z)/2$ 是投影到激发态的投影算符， C 是一个取决于目标激发态特性的常数。量子演化和读出后，会得到由 0 和 1 组成的比特串，分别表示激发态和基态量子比特。为了计算给定可观测量的期望值，需要执行许多轮操作，以生成足够的统计数据。模拟器的输出会提供比特串，用于模拟量子处理器的实际输出。

我们指出，EMU-TN 也可应用于其他哈密顿量，例如可通过中性原子 [60] 和囚禁离子 [61] 阵列构造的长程“XY”哈密顿量。

A. 一维规则晶格：里德伯晶体

哈密顿量（7）的一个显著特性是，在某些参数范围内，两个相邻原子的同时激发会受到抑制。这种现象称为里德伯阻塞 [62, 63], ，当相互作用项 C/r^6 超过拉比频率Ω 时便会发生（因此它取决于距离）。里德伯阻塞机制是中性原子系统中纠缠的主要来源，因为最大纠缠态$|01\rangle + |10\rangle$ 在能量上比乘积态 $|11\rangle$更有利。

里德伯阻塞已被用于 [64] 在中性原子一维链中形成高度规则的激发图样。这些结构称为里德伯晶体；在实验上合理的原子间距和拉比频率下，它们可以被制备成

呈现 $\mathbb{Z}_n$ 序的 $n = 2, 3, 4$。图 3 示意性地描绘了这一情形。

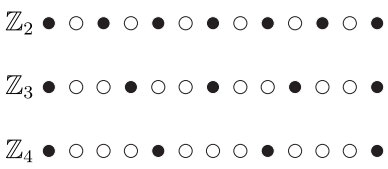


图 3. 由 13 个原子组成的链上的一维里德伯晶体示例。实心圆表示激发态原子，空心圆表示处于基态的原子。 $\mathbb{Z}_2$、$\mathbb{Z}_3$ 和 $\mathbb{Z}_4$ 阶分别以激发之间隔有 1、2 或 3 个基态原子为特征。

我们首先展示由 16 个原子组成的链上的测试模拟结果。对于图 4 上半部分所示的脉冲序列，其中 $\Omega_{\max} = 6.3$ rad $/\mu s$ ，且原子间距为 $4\ \mu m$，我们预计系统的最终基态为 $\mathbb{Z}_3$ 阶。图中还报告了最终态中每个量子比特的激发概率，分别由 EMU-TN和精确薛定谔方程求解器计算得出，见图 4；图中还展示了热图，分别表示精确求解器和EMU-TN 计算得到的每个量子比特激发概率随时间的演化。

接下来，我们展示相同模拟在包含37个原子的链上的结果，而对于该系统，使用精确求解器是不可行的。在这种情况下，我们将脉冲时长设为原来的两倍，以便让关联从边界一直传播到链的中心，并更清晰地观察激发的交替现象。最终态中的激发概率及其时间演化如图5所示。

B. 二维规则晶格：反铁磁态制备

人们通常可以尝试模拟的任务之一，是利用规则寄存器制备特定态。这已被用于可编程中性原子阵列中数百个量子比特的里程碑式实现[5, 6]。

一种典型的脉冲序列（图6，上方）表示热力学极限下相图中的一条路径，最终到达预期反铁磁相中的某一点；该相此前已得到解析研究。我们展示了对演化后 MPS 进行采样的结果，以及在本地计算机上通过数值求解实现的精确对角化结果。使用精确对角化时，可以运行略高于20个量子比特规模的模拟，但随着量子比特数量增加，很快就会变得不切实际。另一方面，通过调整张量网络算法的键维数，可以在相对较短的时间内探索20至60个量子比特范围内序列的行为。此外，簇方法的灵活性还支持任务并行化，从而能够获取有关参数扫描的信息，而无需

用户无需花时间调整自己的代码。在图IVB中，我们列出了在不同数组大小和键维数下，使用和不使用 GPU 时所执行模拟的耗时信息。

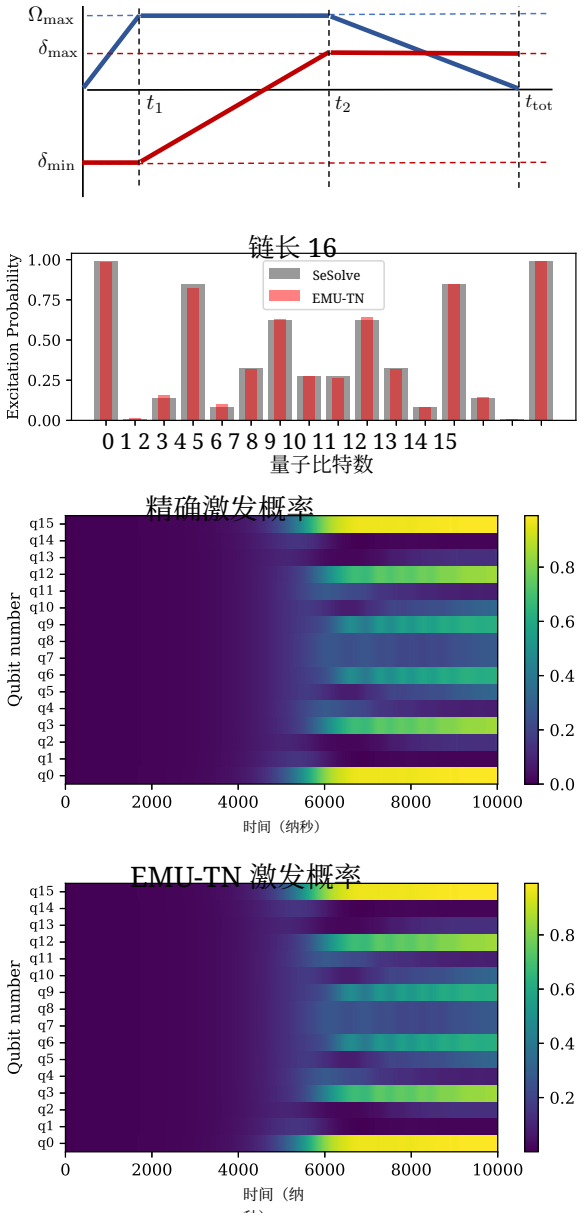


图 4. 使用精确薛定谔方程求解器（使用 QuTiP [65]）与模拟器（EMU-TN）在一维链上制备里德堡晶体的比较。该链由 16 个原子组成，相邻原子间距为 $4\ \mu m$ 。驱动序列表示一个 $t_{tot} = 10\mu s$ 的脉冲，$\Omega_{\max} = 6.3$ 弧度/μs。直方图表示最终态中每个量子比特的激发概率，热图则分别表示使用精确求解器和 EMU-TN 得到的激发概率随时间的演化。

C. 单位圆盘图上的最大独立集

我们介绍了中性原子设备在二维寄存器中的一个典型用例。原子可以放置在任意二维构型中，因此能够在量子处理器上求解某些类别的困难图问题。一项有趣的应用是在单位圆盘（UD）图上求解最大独立集（MIS）问题 [66]。

图是由节点及节点之间的连接构成的对象。若且唯若两个节点之间的距离小于某个最小距离，它们才相连，这样的图称为 UD图。用中性原子表示图中的节点，并使里德伯阻塞

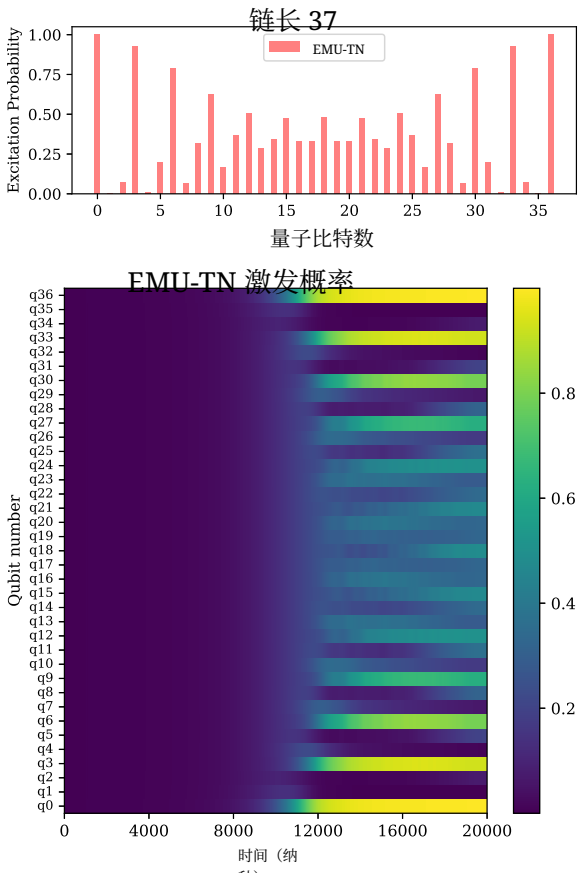


图5。与图4中的模拟相同，但使用一条包含37个量子比特的链，脉冲为 20 μs。

半径与 UD图的最小距离相匹配，就能建立直接映射，使彼此间距小于阻塞距离的原子相互连接。此类系统的量子演化自然受限于希尔伯特空间中禁止相连原子同时激发的那些子空间。这些构型对应于图的独立集，即彼此之间没有直接连接的节点子集。通过驱动量子演化，尽可能地产生更多激发，测量结果就有很高概率得到一个基数较大的独立集，从而为最大独立集问题提供一个较好的候选解。

寻找图的最大独立集的最佳脉冲序列基于图4 和6中已展示的相同绝热脉冲，部分原因在于里德伯晶体和反铁磁态可视为规则一维和二维晶格的最大独立集。如 [67], 所示，不过，对于任意图，通常需要优化脉冲形状，才能以足够高的概率找到最大独立集。我们在图8中展示了一族由单个参数决定的脉冲，其中参数 t_c 控制失谐从负值变为正值的时间。与其依次扫描自由参数，不如针对每个待测试的取值同时提交多个作业，最后选出结果最佳的参数。并行化这项任务是集群基础设施的一项直接优势，而云服务则提供了专用界面，可用于组织和检索所有作业，从而节省大量时间。我们对9个参数进行了扫描，以获得一个

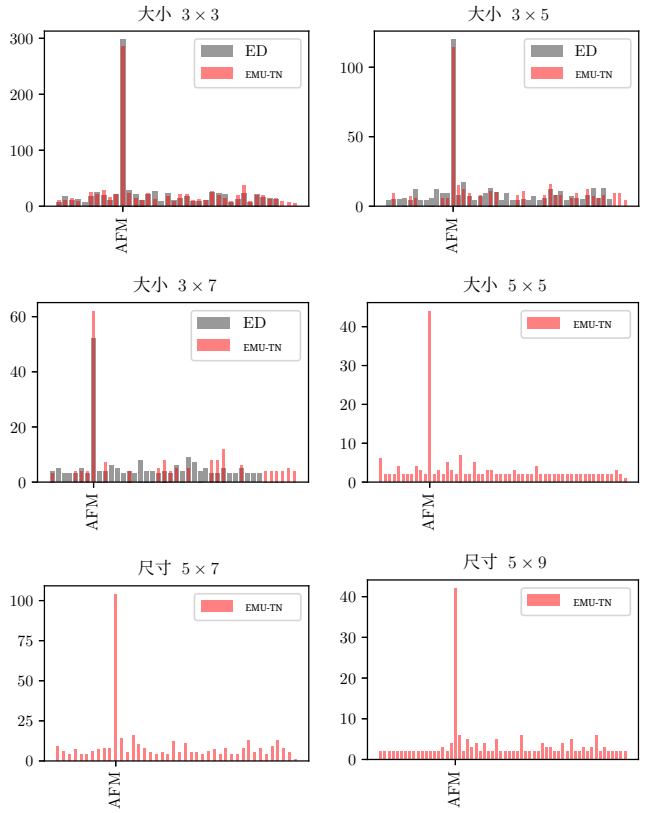


图6。比较精确对角化（10^3 个样本）与模拟器（EMU-TN）对制备的量子态进行采样所得的结果，阵列大小逐渐增加。驱动序列（结构与上文图4相同）表示一个脉冲，$t_{\text{tot}} = 2.7\mu s$ 沿着哈密顿量的相图朝反铁磁相移动。对于 5×7 和 5×9 阵列，需要 10^4 个样本才能更好地分辨峰值（分别为 1% 和 0.4%），因为短脉冲会使大尺寸阵列的绝热性遭到破坏。

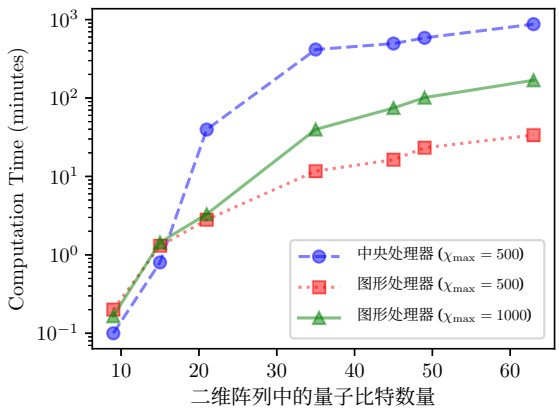


图 7。不同阵列大小下的计算时间。我们仅比较中央处理器的使用情况，以及增加 GPU 后端后的情况，并考察不同最大键维数 $\chi_{\max}$。

在一个含有 30 个节点的图上施加 5 μs 脉冲。结果见下文图8，失谐的最佳过零点约为 1。6 μs。

虽然此处展示的示例侧重于脉冲参数，但也可以实现其他适合并行化的任务，例如对噪声和无序实现取平均值。

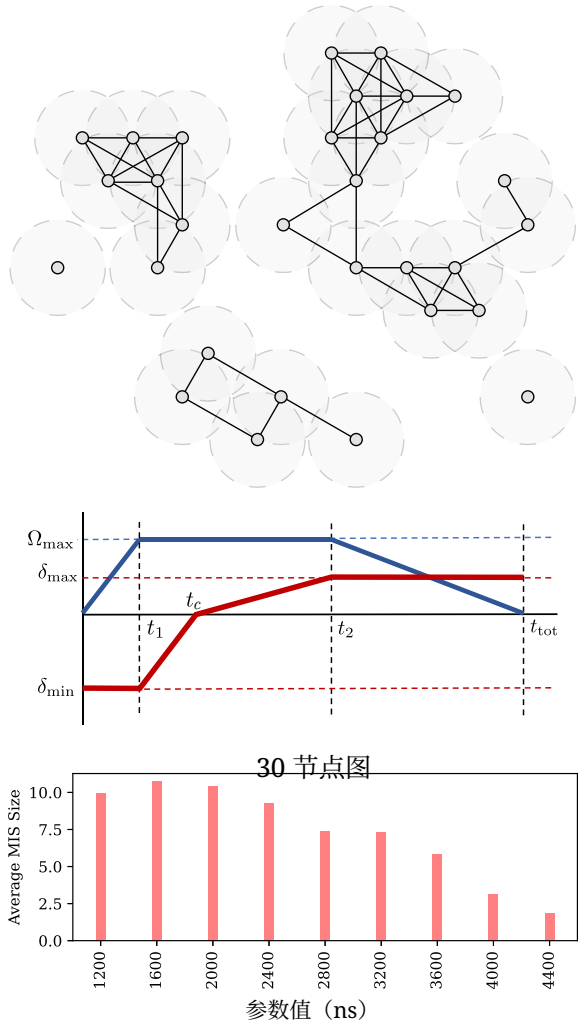


图 8. 参数扫描。我们在二维平面中排列了 30 个量子比特（上图），虚线圆圈表示阻塞半径的一半。脉冲序列（中图，$t_{\text{tot}} = 5\mu s$）包含一个可调参数 t_c，用于控制失谐从负值变为正值的时刻。我们并行扫描了 t_c 的 9 个取值。结果（下图）显示了每个 t_c 取值对应找到的最大独立集的平均大小。整个扫描耗时约 2 小时的云计算时间。

。能够快速为包含许多量子比特的系统搭建并运行此类研究，为兼容量子处理器的创新算法设计打开了大门。值得注意的是，在一维系统中，时间依赖变分原理算法已被用于研究$\mathcal{O}(10^2)$ 自旋 [68, 69]。

V. 讨论

随着人们对开发和使用集成于现代计算基础设施中的量子处理单元（QPU）的兴趣日益浓厚，我们预计将出现需要大量使用模拟器的工作流。模拟器并非只是实际量子设备的替代品，还可用于诊断、资源估算和管理。鉴于当前量子处理单元的价格和可用性，仿真对于应用开发的初期阶段至关重要：它能在研发阶段提供更快的反馈循环，同时还能确保基于给定量子计算架构设置切实可行的约束

。这对于设计量子设备的升级方案也很重要，因为可以快速比较预期行为的相关性能指标。

通过在本地运行的模拟器、大型高性能计算环境中的模拟器以及量子处理单元之间提供无缝接口，研究人员可以灵活选择最合适的计算方案。进一步将这些资源集中到工作流中，并允许调度器管理最优的软件资源，便能同时缩短完成任务所需的总时间并降低成本。工作流还必须考虑计算任务应以最快的墙钟时间、最高能效、最具成本效益的方式执行，还是采用这些目标的加权组合。计算领域在工作流管理方面已取得长足进步，而机器学习的日益广泛应用也催生了机器学习运维（MLOps）等新领域。我们期待量子计算社区致力于构建此类先进工作流，并能够将其集成到现有工作流中，从而获得真正的工业优势。这不仅包括算法的执行，还包括提供适用的训练和测试。

本文介绍了构成此类工作流的基本组件。通过利用能够借助高性能计算集群所提供资源的量子动力学张量网络算法，我们探讨了云按需服务如何利用集群后端，简化对所需计算操作的访问。数值计算可以轻松适配不同的架构和设备，从而将设计工具与量子动力学计算分离开来。研究量子动力学系统的其他技术也同样可以调整应用：未来的目标包括考察低量子比特（< 20）范围，在此范围内，态矢量演化方法能够实现硬件噪声效应的微观模型，并模拟其他状态制备和读出误差 [70]。这包括在算法管理端实现并行化程序、可控地利用集群资源处理繁重的计算负载，以及为其他动力学方程（量子主方程和随机薛定谔方程）开发高效求解器。此外，为低量子比特范围内的常见任务建立一套基准测试，也有助于了解速度瓶颈；这将是后续工作的研究主题。对于较大的阵列，采用投影纠缠对态（PEPS）或树张量网络等张量网络结构，可为二维系统提供其他便捷的表示方式；这些结构在时间演化算法中的应用目前正在积极研究 [71–75]

借助功能完善的云后端来编排任务，工作流的开发会轻松得多。这使得我们能够创建工作流，自动评估提交给量子处理器的脉冲序列是否有效。模拟特定脉冲序列所产生的动力学，需要仔细调整参数，并了解如何设置算法参数。云平台可以根据对模拟的实时监控，为用户提供后续超参数的建议。最后，集群中的工作节点拥有大量本地存储空间，存放在所谓的“暂存”目录中。这使得我们能够以较低成本临时存储大量数据。未来的平台扩展或可允许后续

作业步骤使用先前的结果，将已有计算工作循环用于新任务。

我们相信，此处介绍的平台能够显著提升新兴量子产业研究的可复现性和基准测试水平，同时使更广泛的研究人员、工程师和开发者能够理解、探索并设计量子系统和量子设备的行为。

致谢

我们感谢 Loïc Henriet、Caroline de Groot 和 Matthieu Moreau 提供宝贵的交流和对本文的建议，并在相关工作中与我们合作。

[1] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. 使用可编程超导处理器实现量子霸权。《自然》，574(7779):505–510，2019。[2] Han-Sen Zhong, Hui Wang, Yu-Hao Deng, Ming-Cheng Chen, Li-Chao Peng, Yi-Han Luo, Jian Qin, Dian Wu, Xing Ding, Yi Hu, et al. 利用光子实现量子计算优势。《科学》，370(6523):1460–1463，2020。[3] Yulin Wu, Wan-Su Bao, Sirui Cao, Fusheng Chen, Ming- Cheng Chen, Xiawei Chen, Tung-Hsun Chung, Hui Deng, Yajie Du, Daojin Fan, et al. 利用超导量子处理器实现显著的量子计算优势。《物理评论快报》，127(18):180501，2021。[4] Lars S Madsen, Fabian Laudenbach, Mohsen Falamarzi Askarani, Fabien Rortais, Trevor Vincent, Jacob FF Bul- mer, Filippo M Miatto, Leonhard Neuhaus, Lukas G Helt, Matthew J Collins, et al. 借助可编程光子处理器实现量子计算优势。《自然》，606(7912):75–81，2022。[5]Pascal Scholl, Michael Schuler, Hannah J Williams, Alexander A Eberharter, Daniel Barredo, Kai-Niklas Schymik, Vincent Lienhard, Louis-Paul Henry, Thomas C Lang, Thierry Lahaye, et al. 利用数百个里德伯原子模拟二维反铁磁体。《自然》，595(7866):233–238，2021。[6]Sepehr Ebadi, Tout T Wang, Harry Levine, Alexander Keesling, Giulia Semeghini, Ahmed Omran, Dolev Blu- vstein, Rhine Samajdar, Hannes Pichler, Wen Wei Ho, et al. 在256原子可编程量子模拟器上实现量子物质相。《自然》，595(7866):227– 232，2021。[7] Daniel Jaschke, Alice Pagano, Sebastian Weber, and Si- mone Montangero. 用于量子计算机基准测试的从头算二维数字孪生体。*arXiv* 预印本 *arXiv:2210.03763*，2022。[8] Gian Giacomo Guerreschi, Justin Hogaboam, Fabio Baruffa, and Nicolas PD Sawaya. 英特尔量子模拟器：面向云端的量子电路高性能模拟器。*Quantum Science and Technology*，5(3): 034007，2020。[9] Travis S Humble, Alexander McCaskey, Dmitry I Lyakh, Meenambika Gowrishankar, Albert Frisch, and Thomas Monz. 面向高性能计算的量子计算机。*IEEE Micro*，41(5):15–23，2021。[10] Gokul Subramanian Ravi, Kaitlin N Smith, Pranav Gokhale, and Frederic T Chong. 云端量子计算：分析作业和机器特征。载于*2021 IEEE International Symposium on Workload Characterization (IISWC)*，第39–50页。IEEE，2021。[11]Salvatore Mandrà, Jeffrey Marshall, Eleanor G Rief- fel, and Rupak Biswas. hybridq：一种用于量子电路的混合模拟器。载于*2021 IEEE/ACM Second In-ternational Workshop on Quantum Computing Software (QCS)*，第99–109页。IEEE，2021。[12]Alexander J McCaskey, Eugene F Dumitrescu, Dmitry Liakh, Mengsu Chen, Wu-chun Feng, and Travis S Hum-

。一种与语言和硬件无关的量子—经典计算方法。SoftwareX，7 254，2018 245：–。[13] Ulrich Schollwöck。《矩阵乘积态时代的密度矩阵重整化群》。《物理年鉴》，326(1):96–192，2011年1月。[14]Juan Osorio Iregui、Philippe Corboz 和 Matthias Troyer。使用张量网络算法探究 Kitaev-Heisenberg 模型中自旋液体相的稳定性。《物理评论 B》，90(19):195102，2014。[15] Shenghan Jiang 和 Ying Ran。任意子凝聚与对称保护拓扑相的一般张量网络构造。《物理评论 B》，95(12):125107，2017。[16] Andrzej Cichocki、Namgil Lee、Ivan Oseledets、Anh-Huy Phan、Qibin Zhao 和 Danilo P. Mandic。用于降维和大规模优化的张量网络：第1部分，低秩张量分解。《机器学习基础与趋势》，9(4-5):249–429，2016。doi:10.1561/2200000059。[17] Edwin Stoudenmire 和 David J Schwab。使用张量网络进行监督学习。《神经信息处理系统进展》，29，2016。[18] Feng Pan、Keyang Chen 和 Pan Zhang。求解 Sycamore量子电路的采样问题。《物理评论快报》，129(9):090502，2022。[19] Nicholas J Higham。《矩阵函数：理论与计算》。SIAM，2008。[20] Cleve Moler 和 Charles Van Loan。二十五年后计算矩阵指数的十九种可疑方法。SIAM Review，45(1):3–49，2003。[21] Thomas Barthel 和 Yikang Zhang。针对两个和三个非对易项的优化 Lie–Trotter–Suzuki 分解。《物理年鉴》，418:168165，2020年7月。doi:10.1016/j.aop.2020.168165。URL https://doi.org/10.1016/j.aop.2020.168165。[22] Ernst Hairer、Syvert Paul Nørsett 和 Gerhard Wanner。《常微分方程求解》。I: 非刚性问题。第3次校正版印刷，第8卷。2010。[23] Cristina Cirstoiu、Zoe Holmes、Joseph Iosue、Lukasz Cincio、Patrick J Coles 和 Andrew Sornborger。用于超越相干时间的量子模拟的变分快速前推。npj量子信息，6(1):1–10，2020。[24] Lorenzo Pastori、Tobias Olsacher、Christian Kokail 和 Peter Zoller。通过哈密顿量与刘维尔学习表征和验证 Trotter 化数字量子模拟。arXiv预印本 arXiv:2203.15846，2022。[25] Andrew M Childs、Yuan Su、Minh C Tran、Nathan Wiebe 和 Shuchen Zhu。具有对易子尺度关系的 Trotter 误差理论。《物理评论X》，11 011020，2021 1( )：。[26] Matthias C Caro、Hsin-Yuan Huang、Nicholas Ezzell、Joe Gibbs、Andrew T Sornborger、Lukasz Cincio、Patrick J Coles 和 Zoë Holmes。量子动力学学习的分布外泛化。arXiv预印本 arXiv:2204.10268，2022。[27]Joe Gibbs、Zoë Holmes、Matthias C Caro、Nicholas

Ezzell、Hsin-Yuan Huang、Lukasz Cincio、Andrew T Sornborger 和 Patrick J Coles。通过具有可证明泛化能力的量子机器学习进行动力学模拟。arXiv预印本 arXiv:2204.10269，2022。[28] Filippo Vicentini、Damian Hofmann、Attila Szabó、Dian Wu、Christopher Roth、Clemens Giuliani、Gabriel Pescia、Jannes Nys、Vladimir Vargas-Calderón、Nikita Astrakhantsev 等。NetKet 3：面向多体量子系统的机器学习工具箱。SciPost Physics Codebases，第007页，2022。[29] Hideo Aoki、Naoto Tsuji、Martin Eckstein、Marcus Kollar、Takashi Oka 和 Philipp Werner。非平衡动力学平均场理论及其应用。Reviews of Modern Physics，86(2):779，2014。[30] Inés De Vega 和 Daniel Alonso。非马尔可夫开放量子系统的动力学。Reviews of Modern Physics，89(1):015001，2017。[31] Martin B Plenio 和 Peter L Knight。量子光学中耗散动力学的量子跳跃方法。Reviews of Modern Physics，70(1):101，1998。[32] Andrzej Cichocki、Namgil Lee、Ivan Oseledets、Anh-Huy Phan、Qibin Zhao、Masashi Sugiyama 和 Danilo P. Mandic。用于降维和大规模优化的张量网络：第2部分，应用与未来展望。《机器学习基础与趋势》，9(6):249–429，2017。doi:10.1561/2200000067。URL https://doi.org/10.1561/2200000067。[33] Jacob C Bridgeman 和 Christopher T Chubb。信手涂鸦与诠释性舞蹈：张量网络入门课程。《物理学杂志 A：数学与理论》，50(22):223001，2017。[34] José I Latorre。纠缠熵与量子力学模拟。《物理学杂志 A：数学与理论》，40(25):6689，2007。[35] F. Verstraete 和 J. I. Cirac。用于二维及更高维量子多体系统的重整化算法，2004。URL https://arxiv.org/abs/cond-mat/0407066。[36] Y.-Y. Shi、L.-M. Duan 和 G. Vidal。使用树张量网络对量子多体系统进行经典模拟。《物理评论 A》，74(2)，2006年8月。doi: 10.1103/physreva.74.022320。URL https://doi.org/10.1103/physreva.74.022320。[37] L. Tagliacozzo、G. Evenbly 和 G. Vidal。利用纠缠面积定律的树张量网络模拟二维量子系统。《物理评论 B》，80(23)，2009年12月。doi: 10.1103/physrevb.80.235127。URL https://doi.org/10.1103/physrevb.80.235127。[38]Román Orús。张量网络实用入门：矩阵乘积态与投影纠缠对态。《物理年鉴》，349:117–158，2014。ISSN 0003-4916。[39] Michael P. Zaletel、Roger S. K. Mong、Christoph Karrasch、Joel E. Moore 和 Frank Pollmann。具有长程相互作用的矩阵乘积态时间演化。《物理评论 B》，91(16)，2015年4月。doi: 10.1103/physrevb.91.165112。URL https://doi.org/10.1103/physrevb.91.165112。[40] Sebastian Paeckel、Thomas Köhler、Andreas Swoboda、Salvatore R. Manmana、Ulrich Schollwöck 和 Claudius Hubig。矩阵乘积态的时间演化方法。《物理年鉴》，411:167998，2019年12月。[41] I. V. Oseledets。张量列分解。SIAM科学计算杂志，33(5):2295–2317，2011年1月。[42] Jutho Haegeman、Christian Lubich、Ivan Oseledets、Bart Vandereycken 和 Frank Verstraete。通过矩阵乘积态统一时间演化与优化。《物理评论 B》，94(16):165116，2016。[43] Julian David Teske、Pascal Cerfontaine 和 Hendrik Bluhm。qopt：一款面向实验的量子比特模拟与量子最优控制软件包。《物理评论应用》，17(3):034036，2022。[44] Boxi Li、Shahnawaz Ahmed、Sidhant Saraogi、Neill Lam-bert、Franco Nori、Alexander Pitchford 和 Nathan Shammah。使用 qutip 实现脉冲级噪声量子电路。Quantum，6:630，2022。[45]Thomas Alexander、Naoki Kanazawa、Daniel J Egger、Lauren Capelluto、Christopher J Wood、Ali Javadi-Abhari 和 David C McKay。Qiskit pulse：通过云端脉冲对量子计算机进行编程。Quantum Science and Technology，5(4):044006，2020。[46] Henrique Silvério、Sebastián Grijalva、Constantin Dalyac、Lucas Leclerc、Peter J Karalekas、Nathan Shammah、Mourad Beji、Louis-Paul Henry 和 Loïc Henriet。Pulser：一款用于设计可编程中性原子阵列脉冲序列的开源软件包。Quantum，6 629，2022：。[47]Marco Rossignolo、Thomas Reisser、Alastair Marshall、Phila Rembold、Alice Pagano、Philipp J Vetter、Ressa S Said、Matthias M Müller、Felix Motzoi、Tommaso Calarco 等。QuOCS：量子最优控制套件。arXiv 预印本 arXiv:2212.11144，2022。[48] Michael H Goerz、Sebastián C Carrasco 和 Vladimir S Malinovsky。通过半自动微分实现量子最优控制。arXiv 预印本 arXiv:2205.15044，2022。[49] 云原生计算基金会。Kubernetes。网址：https://kubernetes.io/。[50] Andy B. Yoo、Morris A. Jette 和 Mark Grondona。Slurm：简单的 Linux 资源管理工具。载于 Dror Feitelson、Larry Rudolph 和 Uwe Schwiegelshohn 编，《并行处理作业调度策略》，第 44–60 页，柏林、海德堡，2003。Springer Berlin Heidelberg。ISBN 978-3-540-39727-4。[51] Tim Wickberg。Slurm 和/或/与 Kubernetes。SuperComputing，2022。网址：
h
t
tps://slur
m.schedmd.com/SC22/Slurm-and-or-vs-Kubernetes.pdf。[52] Naweiluo Zhou、Yiannis Georgiou、Li Zhong、Huan Zhou 和 Marcin Pospieszny。高性能计算系统上的容器编排，2020。网址：https://arxiv.org/abs/2012.08866。[53] Boris Lublinsky、Elise Jennings 和 Viktória Spišaková。连接云与外部资源的 Kubernetes“桥接”运算符，2022。网址：https://arxiv.org/abs/2207.02531。[54] Tim Costa。NVIDIA QODA：量子-经典混合计算平台。网址：https://developer.nvidia.com/qoda/。[55] Nate Rini。REST API。Slurm 用户组会议，2020。网址：https://slurm.schedmd.com/SLUG20/REST_ API.pdf。[56] PASQAL 云服务。Pasqal 云 SDK。https://github.com/pasqal-io/cloud-sdk，2021。[57] Loïc Henriet、Lucas Beguin、Adrien Signoles、Thierry Lahaye、Antoine Browaeys、Georges-Olivier Reymond 和 Christophe Jurczak。使用中性原子进行量子计算。Quantum，4:327，2020。[58] Mark Saffman、Thad G Walker 和 Klaus Mølmer。利用里德堡原子实现量子信息。Reviews of Modern Physics，82(3):2313，2010。[59] M Morgado 和 S Whitlock。利用里德堡相互作用量子比特进行量子模拟与计算。AVS Quantum Science，3(2):023501，2021。[60] Sylvain de Léséleuc、Daniel Barredo、Vincent Lienhard、Antoine Browaeys 和 Thierry Lahaye。对里德堡原子间共振偶极-偶极相互作用的光学控制。《物理评论快报》，119(5):053202，2017。[61] Petar Jurcevic、Ben P Lanyon、Philipp Hauke、Cornelius

Hempel、Peter Zoller、Rainer Blatt 和 Christian F Roos。在量子多体系统中进行准粒子工程与纠缠传播。《自然》，511 (7508):202–205，2014。[62] D. Jaksch、J. I. Cirac、P. Zoller、S. L. Rolston、R. Côté 和 M. D. Lukin。面向中性原子的快速量子门。《物理评论快报》，85(10):2208–2211，2000 年 9 月。doi:10.1103/physrevlett.85.2208。网址：https://doi.org/10.1103/physrevlett.85.2208。[63] Antoine Browaeys 和 Thierry Lahaye。利用单独控制的里德堡原子研究多体物理。《自然·物理》，16(2):132–142，2020 年 1 月。doi:10.1038/s41567-019-0733-z。网址：https://doi.org/10.1038/s41567-019-0733-z。[64] Hannes Bernien、Sylvain Schwartz、Alexander Keesling、Harry Levine、Ahmed Omran、Hannes Pichler、Soon-won Choi、Alexander S. Zibrov、Manuel Endres、Markus Greiner、Vladan Vuletić 和 Mikhail D. Lukin。在含 51 个原子的量子模拟器中探测多体动力学。《自然》，551(7682):579–584，2017 年 11 月。ISSN 1476-4687。[65] J Robert Johansson、Paul D Nation 和 Franco Nori。Qutip：一款用于开放量子系统动力学的开源 Python 框架。Computer Physics Communications，183(8):1760–1772，2012。[66] Hannes Pichler、Sheng-Tao Wang、Leo Zhou、Soonwon Choi 和 Mikhail D. Lukin。利用里德堡原子阵列求解最大独立集的量子优化。2018。doi:10.48550/ARXIV.1808.10816。网址：https://arxiv.org/abs/1808.10816。[67] S. Ebadi、A. Keesling、M. Cain、T. T. Wang、H. Levine、D. Bluvstein、G. Semeghini、A. Omran、J.-G. Liu、R. Samajdar、X.-Z. Luo、B. Nash、X. Gao、B. Barak、E. Farhi、S. Sachdev、N. Gemelke、L. Zhou、S. Choi、H. Pichler、S.-T. Wang、M. Greiner、V. Vuletić 和 M. D. Lukin。利用里德堡原子阵列求解最大独立集的量子优化。《科学》，376(6598):1209–1215，2022。doi:10.1126/science.abo6587。网址：https://www.science.org/doi/abs/10.1126/science.abo6587。[68] Mingru Yang 和 Steven R White。采用辅助 Krylov 子空间的含时变分原理。《物理评论 B》，102(9):094315，2020。[69] Jheng-Wei Li、Andreas Gleis 和 Jan Von Delft。矩阵乘积态的受控键扩展含时变分原理。arXiv 预印本 arXiv:2208.10972，2022。[70] Sylvain De Léséleuc、Daniel Barredo、Vincent Lienhard、Antoine Browaeys 和 Thierry Lahaye。单个原子相干光学激发至里德堡态过程中的缺陷分析。《物理评论 A》，97(5):053803，2018。[71] Michael P Zaletel、Roger SK Mong、Christoph Karrasch、Joel E Moore 和 Frank Pollmann。具有长程相互作用的矩阵乘积态时间演化。《物理评论 B》，91(16):165112，2015。[72] Daniel Bauernfeind 和 Markus Aichhorn。树张量网络的含时变分原理。SciPost Physics，8(2):024，2020。[73] Marek M Rams 和 Michael Zwolak。突破纠缠壁垒：量子输运的张量网络模拟。《物理评论快报》，124(13):137701，2020。[74] Benedikt Kloss、David Reichman 和 Yevgeny Bar Lev。利用树张量网络态研究二维量子晶格中的动力学。SciPost Physics，9(5):070，2020。[75] Bram Vanhecke、Laurens Vanderstraeten 和 Frank Verstraete。基于张量网络的对称簇展开。《物理评论 A》，103(2):L020402，2021。[76] Matthew Fishman、Steven R. White 和 E. Miles Stoudenmire。用于张量网络计算的 ITensor 软件库。SciPost Phys. Codebases，第 4 页，2022。doi:10.21468/SciPostPhysCodeb.4。URL https://scipost.org/10.21468/SciPostPhysCodeb.4。[77] Matthew Fishman 和 Miles Stoudenmire。ITensorTDVP。https://github.com/ITensor/ITensorTDVP.jl，2022。[78] Jeff Bezanson、Alan Edelman、Stefan Karpinski 和 Viral B Shah。Julia：一种数值计算的新方法。*SIAM Review*，59(1):65–98，2017。URL https://doi.org/10.1137/141000671。

附录

A. 计算集群

作为现代高性能计算集群基础设施的一个示例——预计将在其中执行正文提及的算法——我们介绍 PASQAL 的计算集群。请注意，不同提供商的计算能力差异很大，其可用性和成本也各不相同。在需要执行密集型计算任务时，这是一个必须考虑的关键因素。

中央处理器和 GPU 都可用于计算复杂的科学算法，不过它们通常用于不同类型的任务。中央处理器通常用途更广，可处理各种任务，包括运行操作系统和其他软件、管理内存和输入/输出，以及执行复杂计算。它们非常适合需要顺序处理的任务，例如按特定顺序执行指令或遵循一组规则。另一方面，GPU 是专用处理器，专为处理渲染图像和视频所需的大量计算而设计。它们尤其适合可并行化的任务，也就是说，这些任务可以拆分成更小的部分，每部分执行相同的操作，但处理的数据可能不同。

GPU.- 英伟达 NVIDIA A100 图形处理器可为人工智能、数据分析和高性能计算（HPC）等众多应用带来显著加速。高度并行的算法可高效扩展至数千个 GPU；借助英伟达多实例图形处理器（MIG）技术，也可将其划分为七个 GPU 实例，从而加速各种规模的工作负载。此外，第三代张量核心可为各种工作负载加速所有精度的计算。目前，PASQAL 拥有 80 个 A100 英伟达 GPU，分布在 10 台英伟达 DGX A100 系统中。这些服务器通过 Mellanox 专有的 200 Gbps 以太网光纤连接彼此相连。每台 DGX A100 系统由 2 个双路 64 核 AMD EPYC 7742 中央处理器和 8 个 NVIDIA A100 SXM4 GPU 组成。每个 GPU 通过 NVLINK 互相连接，为单个节点提供 320 GB 的共享 GPU 内存池，此外，AMD 中央处理器还配有 1 TB 的 RAM 内存。

*CPU.-*配备两颗 64 核 EPYC CPU 和 1TB 系统内存的 DGX A100，即使不考虑 GPU，也拥有出色的性能。AMD EPYC“Rome”CPU 的架构不在本文讨论范围内，但其设计本身十分精巧。每颗 CPU 均提供 64 个处理器核心（最多支持 128 个线程）、256MB L3 缓存以及八个通道

DDR4-3200 内存（其内存吞吐量在所有主流 x86 CPU 中最高）。15TB 高速 NVMe 固态硬盘临时存储空间可实现极快的数据读写速度；当计算过程中需要在不同步骤将大量数据从磁盘载入 RAM 内存时，这无疑会影响整体处理速度。除节点上的临时存储空间外，该集群还配备了 15TB 分布式文件系统，用于存放用户主目录、软件和容器。我们通过 Slurm 分区，将本文所述云服务带来的内部、外部和生产环境使用情况区分开来。

B. TDVP 实现细节

在第 IV 节所示的应用中，张量网络计算通过使用以 Julia 语言编写的 ITensor.jl [76] 和 ITensorTDVP.jl [77] 软件包实现 [78]。我们选择两位点含时变分原理作为时间演化算法，时间步长为 $\delta t = 10.0$ ns。为处理时间演化过程中的纠缠增长，我们采用自适应键维数 χ_σ，其默认最大值为 $\chi^{\max} = 400$，并可根据具体问题的需求进行调整。每个 $A[\sigma]$时刻的 χ_σ 实际值由统一截断参数 ϵ 控制，使得 SDV 分解中被舍弃的奇异值 $\sum_{i\in\text{被舍弃}} \lambda_i^2 < \epsilon$。我们的设置允许用户在三个压缩级别 $\epsilon \in \{10^{-8}, 10^{-10}, 10^{-12}\}$之间进行选择。