CN 43-1258/TP
ISSN 1007-130X

计算机工程与科学
Computer Engineering & Science

第 43 卷第 1 期 2021 年 1 月
Vol. 43, No. 1, Jan. 2021

文章编号:1007-130X(2021)01-0017-07

量子线路模拟器 QuEST 在多 GPU 平台上的性能优化*

张　亮,常　旭,秦志楷,沈　立
(国防科技大学计算机学院,湖南 长沙 410073)

摘　要:在当前量子计算的研究中,量子线路模拟器作为重要的研究工具,一直受到研究者们的高度重视。QuEST 是一款开源的通用量子线路模拟器,能在单个 CPU 结点、多个 CPU 结点和单个 GPU 等多种测试平台上灵活运行。量子线路模拟固有的并行性使其非常适合在 GPU 上运行,并能获得较大的性能加速。但是其缺点在于所消耗的内存空间巨大,单个 GPU 受显存容量限制,无法模拟具有更多量子位的量子系统。设计并实现了多 GPU 版本的 QuEST 模拟器,解决了单个 GPU 显存不足的问题,能够使用多个 GPU 模拟更多的量子位。而且,与单 CPU 版本相比可获得 7～9 倍的性能加速,与多 CPU 版本相比可获得 3 倍的性能加速。

关键词:量子计算;量子线路模拟器;QuEST;多 GPU;显存
中图分类号:TP393　　**文献标志码**:A
doi:10.3969/j.issn.1007-130X.2021.01.003

Performance optimization of quantum circuit simulator QuEST on multi-GPU platform

ZHANG Liang, CHANG Xu, QIN Zhi-kai, SHEN Li
(College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China)

Abstract: In the current quantum computing research, as an important research tool, quantum circuit simulators have always been highly valued by researchers. QuEST is an open source general-purpose quantum circuit simulator that can run flexibly on multiple test platforms such as a single CPU node, multiple CPU nodes, and a single GPU. The inherent parallelism of quantum circuit simulator makes it very suitable for running on the GPU, and obtain greater performance acceleration. However, the disadvantage is that the memory space consumed is huge. A single GPU is limited by the memory capacity and cannot simulate a quantum system with more qubits. This paper designs and implements a multi-GPU version of the QuEST simulator, which solves the problem of insufficient memory of single GPU and can simulate more qubits. Moreover, compared with the single CPU version, it can achieve 7～9x performance acceleration, and compared with the multi-CPU version, it achieves 3x performance acceleration.

Key words: quantum computing; quantum circuit simulator; QuEST; multi-GPU; GPU memory

1　引言

量子计算是一种全新的计算模式,它以量子位为信息单元,遵循量子力学的规律,通过调控量子信息单元来进行计算[1]。量子计算机以量子力学为基础,其理论模型是量子图灵机。由于量子力学态存在叠加原理,量子信息的处理更加迅速,与通用电子计算机相比具有更强大的信息处理能力[2],在合适的量子算法支持下,能够以很低的时间开销

* 收稿日期:2020-06-11;修回日期:2020-07-15
基金项目:国家自然科学基金(61972407)
通信地址:410073 湖南省长沙市国防科技大学计算机学院
Address: College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, Hunan, P. R. China

来完成密码学、多体系统量子力学和量子机器学习[3]等领域的诸多复杂算法。

然而,量子计算机的实现非常困难,目前仍没有实用化的量子计算机可以用来求解大规模的科学计算问题[4],因此使用电子计算机来模拟量子计算系统成为最常用的研究手段。许多量子线路模拟器应运而生,用来支撑量子领域复杂计算问题的模拟,如 ProjectQ、qHiPSTER 和 QuEST 等。这些模拟器作为重要的研究工具,受到了量子计算研究者的高度重视。但是,目前开发出的量子线路模拟器往往会消耗更多的存储资源,带来更多的计算成本。

ProjectQ[5]是一款用于模拟量子计算的开源软件,它的初始版本就具有针对不同硬件的编译器框架,该框架可以对量子算法进行仿真测试,并可以连接到 IBM Quantum Experience 云服务的后端,在已有的量子硬件上运行。

qHiPSTER[6]是一款在电子计算机上实现的高性能分布式量子线路模拟器,可以模拟常规的单量子位门和双量子位控制门,并针对单 CPU 和多 CPU 做出了优化,包括向量化、多线程和高速缓存等。该模拟器既可以实现高性能模拟,又可以达到较高的硬件利用率,仅受到内存的限制[7]。

QuEST 量子线路模拟器[8]的全称为量子精确模拟工具包(the Quantum Exact Simulation Toolkit),是第 1 款开源、采用 OpenMP 和 MPI 混合编程开发、支持单 GPU 加速的通用量子线路模拟器;从日常生活中的笔记本电脑到世界上功能最强大的超级计算机,它在这些经典计算平台上都可以实现量子计算的模拟。QuEST 能够在纯态和混合态上、在有退相干的情况下,用状态向量和密度矩阵表示量子状态,模拟由一般的单量子位门、双量子位门和多量子位控制门组成的通用量子线路[9]。

虽然目前已经有了几款开源的量子线路模拟器,但它们普遍都存在资源开销大、模拟时间长等问题,而且不能很好地利用 GPU 等计算加速器,有的完全没有 GPU 版本,有的只能在单 GPU[10]上运行,有的虽然有多 GPU 版本,但不具有扩展性和多平台移植性[11]。本文以 QuEST 为例,分析了其代码结构和性能特性,并实现了 QuEST 模拟器在多 GPU 平台上的性能优化。测试结果表明,在 4 块 NIVIDIA V100 GPU 构成的平台上,相较于单 CPU 版本可获得 7~9 倍的性能加速,相较于多 CPU 版本可获得 3 倍的性能加速。

2 QuEST 分析

在量子系统中,量子状态不仅可以是二进制 1 或 0,而且可以是同时持有 1 和 0 的多种组合,从而形成一个“量子位”。如式(1)和式(2)所示,一个量子位的状态可以用一个二维的状态向量来表示,其中 α 和 β 都是复数。多个量子位的表示同理,只要增加状态向量的维数即可。

$$|\varphi\rangle=\alpha|0\rangle+\beta|1\rangle \tag{1}$$

$$|\varphi\rangle=\begin{bmatrix}\alpha\\ \beta\end{bmatrix} \tag{2}$$

量子门是在一组量子位之间进行操作的基本量子线路,是量子线路的基础[12]。在进行量子线路模拟时,多个量子门的运算是串行完成的,每个量子门都会对全部的状态向量进行一次遍历,完成状态向量的更新计算。QuEST 模拟器实现了 tGate、sGate、hadamard、RotateX/Y/Z 和 PauliX/Y/Z 等 14 种量子门。

2.1 算法分析

QuEST 模拟器记录了被模拟量子系统的状态,以及各个进程的相关信息。它用一个复数数组来存放全部状态向量,每个状态向量由 2 个 double 类型的浮点数构成,分别表示状态向量的实部和虚部,是 QuEST 模拟器进行计算的基本数据单元。状态向量的总数与被模拟的量子位数有关,假设对拥有 N 个量子位的量子系统进行模拟,就需要使用 2^N 个状态向量来进行表示。例如,当 $N=30$ 时,2^N 个状态向量共需要 16 GB 内存空间。被模拟的量子系统每增加 1 个量子位,状态向量需要的存储空间就会随之翻 1 倍,可见 QuEST 模拟器的内存消耗很大。

为了提高模拟速度,QuEST 模拟器实现了 CPU 分布式版本,可以使用多个进程在 CPU 平台上进行量子系统的并行模拟。在并行模拟的初始化阶段会对状态向量分组,组数与进程数相同,每个进程维护 1 组私有状态向量,并负责它们的更新。对于分布式版本,每个进程还需要 1 个额外的状态向量数组,专门用于进程间的数据通信,其大小与进程私有的状态向量数组相同,这就导致使用多个进程进行并行量子线路模拟时,存储空间消耗更加巨大。

QuEST 源码的核心部分是实现了 14 个基本量子门的模拟函数,这些模拟函数是基于 2 个模板实现的。tGate、sGate 和 PauliZ 使用了同一个模

板,其特点在于根据输入参数来决定要更新的数据,不存在进程间数据交换。其余函数使用另一种模板,根据输入参数找到对应的进程,先进行数据交换,再完成数据更新。

函数 *statevec_compactUnitary*()是 RotateX/Y/Z、PauliX/Y 等量子门的核心函数,下面以该函数为例来分析 QuEST 模拟器的核心计算过程,找出其性能上的不足。

该函数的工作是先将所有状态向量划分为 2 部分,索引标记分别为 a 和 b,二者之间的差值由输入参数 t 决定(满足关系:$2^t=a-b$),然后利用 a 与 b 对应的状态向量值,使用规定的算法进行更新后得到新的状态向量值,每个状态向量值的更新都需要同时使用到 a 与 b 对应的向量数据。

当使用多个进程进行分布式模拟时,如果 a 与 b 对应的状态向量属于不同进程,那么先要找到与本地对应的进程(这个过程称为进程配对),配对的进程间进行通信,将对方的状态向量拷贝到本地,然后再继续状态向量的更新工作;如果 a 与 b 对应的状态向量都在本地,那么就不需要配对和通信,直接进行状态向量的更新工作。该计算过程的时间开销很大,原因在于每个门函数的计算都需要将所有状态向量遍历 1 次,数据局部性很差;同时,进程间的通信也会带来很大的时间开销。

值得一提的是,在执行具体的门函数时,同一门函数的核心计算部分循环内部并没有数据相关,可以将循环完全并行化处理,这正是使用单 GPU 版本可以获得明显加速效果的原因。但是,由于显存限制,单 GPU 版本并不能满足量子位更多的(例如 $N>30$ 时)量子系统的模拟。

2.2　QuEST 的性能特征

通过对 QuEST 源码的算法分析,可以发现 QuEST 是一款计算密集型的应用软件,具有良好的可扩展性和并行性,并且内存消耗巨大。本节以 30 个量子位的量子系统为例,使用 Random 与 GHZ_QFT 2 个量子线路作为测试算例,分别选取单 CPU、多 CPU 和单 GPU 这几种不同的测试平台,对 QuEST 模拟器的性能进行测试。

Random 测试算例是随机生成的,从 Random 测试算例的部分代码中可以看出,Random 程序调用的门函数是无序、随机的,而 GHZ_QFT 测试算例模拟的是实际量子线路。

Random 测试算例的部分代码如下所示:

```
Qureg q=createQureg(30,Env);
  float q_measure[30];
  tGate(q,25);
  controlledNot(q,28,21);
  controlledRotateX(q,17,5,0.3293660327520663);
  rotateX(q,27,3.9207427542347926);
  tGate(q,3);
  controlledRotateZ(q,27,19,5.459935259485407);
  controlledRotateX(q,26,3,4.736990305652013);
  controlledRotateZ(q,8,11,3.594728080156504);
  rotateX(q,10,4.734238389048838);
  rotateY(q,8,4.959946047271496);
  rotateZ(q,5,1.0427019597472071);
  ⋮
```

对比 Random 测试算例的代码与 GHZ_QFT 测试算例的代码可以发现,GHZ_QFT 程序中的门函数更加规整,相较于 Random 测试算例更具有实际的研究意义。

GHZ_QFT 测试算例的部分代码如下所示:

```
/* QFT starts */
hadamard(q,0);
controlledRotateZ(q,0,1,1.5708);
hadamard(q,1);
controlledRotateZ(q,0,2,0.785398);
controlledRotateZ(q,1,2,1.5708);
hadamard(q,2);
controlledRotateZ(q,0,3,0.392699);
⋮
```

测试环境如表 1 所示,每个计算结点有 2 块 Intel Xeon E5-2692 V2 处理器(内存 64 GB)和 2 块 GPU。GPU 分别选用 NVIDIA Tesla K80(显存 12 GB)与 NVIDIA Tesla V100(显存 32 GB)2 种型号。测试结果分别如图 1 与图 2 所示。

Table 1　Hardware and environment of test experiment

表 1　测试实验的硬件配置与运行环境

项目		配置
操作系统		Red Hat Enterprise Linux Server release 6.5
硬件配置	CPU	Intel Xeon E5-2692V2@2.2 GHz*8
	内存	64 GB
	网络	TH Express-2
	GPU	NVIDIA Tesla V100 32 GB NVIDIA Tesla K80 12 GB
运行环境	源码	QuEST-2.1.0
	编译器	GNU-4.4.7
	并行环境	OpenMPI-1.8.8
	CUDA	CUDA-9.2

图 1 中横轴表示每组测试所使用的计算平台,纵轴表示每组测试的运行时间,单位为秒。

图 2 中横轴表示每组测试的加速比(以单

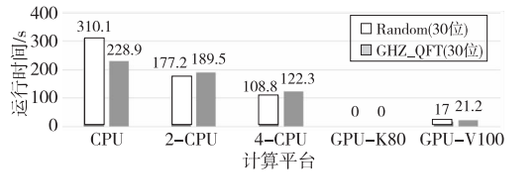


Figure 1 QUEST simulation time (30 bits)

图 1 30 位 QUEST 模拟时间

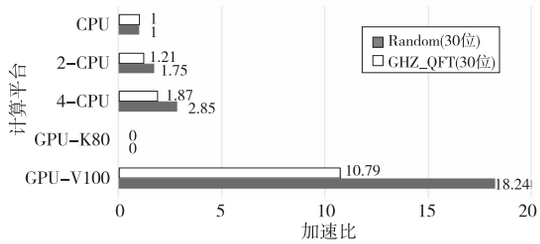


Figure 2 QUEST performance speedup (30 bits)

图 2 30 位 QUEST 性能加速比

CPU 性能为基准),纵轴表示每组测试所使用的计算平台。

测试结果表明,对比单 CPU 与多 CPU 版本代码的运行时间,可以确认 QuEST 可扩展性良好。受 GPU 内存限制,30 个量子位的量子系统无法在显存仅有 12 GB 的 NVIDIA Tesla K80 上完成模拟,而使用 32 GB 显存的 NVIDIA Tesla V100 运行单 GPU 版本,相较于单 CPU 版本可以获得高达 10.79 倍和 18.24 倍的性能加速比,加速效果明显。这说明 QuEST 的并行性极好,十分契合 GPU 的单指令多线程体系结构,使用 GPU 设备来模拟运行再合适不过,但是对单个 GPU 的显存容量要求很高。因此,本文设计并实现了 QuEST 模拟器的多 GPU 版本,突破了单个 GPU 内存容量对 QuEST 模拟器的限制,并大大提高了模拟速度。

3 多 GPU 版本的实现

3.1 总体框架

1 个计算结点通常包含多个 GPU 设备,QuEST 模拟器在 MPI 启动时可以指定结点数和每个结点的进程数,如果采用每个 GPU 设备绑定 1 个 MPI 进程的设置,当程序处理的数据规模变大,需要增加 GPU 时,只需要在程序启动时修改 MPI 的初始化参数即可,这样程序就具有较强的可扩展性。反之,如果每个结点绑定 1 个 MPI 进程,即 1 个 MPI 进程管理多个 GPU 设备,程序的可扩展性就会大幅度下降。当结点内的 GPU 数量变化时,需要对应地修改程序,程序的可扩展性较差。

本文对 QuEST 的单 GPU 实现进行了修改。首先为每个 GPU 设备绑定 1 个 MPI 进程;其次在初始化阶段根据模拟的量子位数为 GPU 分配相应的内存空间,状态向量按照从前至后的顺序分配到多个 GPU 设备上;然后将单 GPU 版本的量子门模拟函数修改为多 GPU 版本;最后完成多 GPU 之间的数据规约操作,得到正确的计算结果。

门函数的多 GPU 版本实现参考了 CPU 分布式版本的代码框架,这里仍旧以 *statevec_compactUnitary*() 这个核心计算函数为例,当使用多个 GPU 时,初始化阶段完成状态向量的分组,组数与进程数一致;核心函数将状态向量分为 α 与 β 2 部分,并判断 2 部分数据是否属于同一进程;若是,直接进行状态向量的更新计算;若否,进程配对后进行 GPU 间通信,将彼此的数据拷贝到对方的 GPU 显存中,然后进行更新计算。其他门函数的实现过程与此类似,不再赘述。

3.2 多 GPU 通信

使用多个 GPU 并行执行应用程序时,需要合理地选择 GPU 间的通信方式(或组合),这对 GPU 间的数据传输效率具有非常大的影响[13]。多 GPU 系统提供了 2 种 GPU 连接方式:(1)单个结点内的所有 GPU 连接到该结点的 PCIe 总线上;(2)结点间的多个 GPU 通过集群中的网络交换机连接在一起。这 2 种连接方式并不互斥,其拓扑结构如图 3 所示。

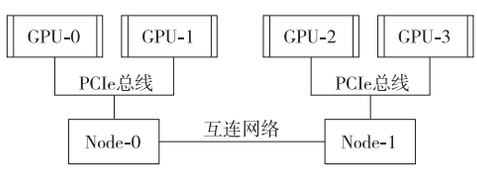


Figure 3 Topology of multi-GPU system

图 3 多 GPU 系统的拓扑结构

点对点(P2P)传输可以实现 GPU 设备间的直接通信,对于连接在同一条 PCIe 总线上的多个 GPU 而言,这要比使用主机内存中转的通信方式高效得多。不同结点上的 GPU 设备也可以通过互连网络来完成点对点通信,现有的 MPI 库支持 CUDA-aware,实现了对 GPU 显存的直接访问。

由于采取 1 个进程绑定 1 个 GPU 的设计思想,所以进程间的通信问题就转换为 GPU 间的数据交换问题。为每个进程设置了 2 个数组:一个用来存放本地的状态向量数据;另一个用来存放从配对进程拷贝过来的状态向量数据。并直接使用

MPI 库中的消息传递函数,在互相配对的 2 个进程间传输向量数据,以这样的方式来实现多个 GPU 间的通信。

此外,在实际对源码修改的过程中,发现 MPI 库中的消息传递函数无法使用"指针+偏移量"的方式来直接访问 GPU 设备上的数据,因而无法在 GPU 间分段传输数据,因此本文做出了进一步改进,使用"以空间换时间"的方法解决了此问题。具体步骤为:(1)为每一个进程再分配 2 个缓冲区 $B1$ 和 $B2$,用于完成 GPU 间的消息传递,其中,$B1$ 用于向配对进程发送数据,$B2$ 用于接收来自配对进程的数据;(2)实现 2 个 Kernel 函数,一个将数组内的部分数据写入到 $B1$ 内,另一个将 $B2$ 内的数据写回到数组内,从而实现了本地与这 2 个缓冲区之间数据的分段传递。

4　性能测试与分析

本节仍然采用与第 2 节相同的环境和算例进行性能测试。

4.1　模拟时间

本节首先对 30 个量子位的量子系统设计了一组测试实验,对多 GPU 版本的模拟时间进行测试。仍使用 Random 与 GHZ_QFT 2 个测试算例,分别在单 CPU、多 CPU、单 GPU 和多 GPU 4 个平台上进行测试,多 CPU 和多 GPU 的数量均为 4,实验结果分别如图 4 与图 5 所示。

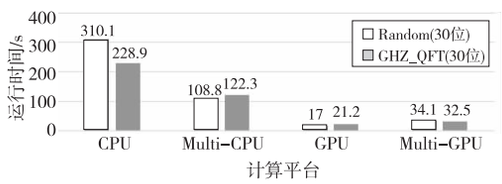


Figure 4　Comparison of simulation time between multi-GPU version and other versions (30 bits)

图 4　多 GPU 版本与其它版本模拟时间比较(30 位)

图 4 中横轴表示每组测试所使用的计算平台,纵轴表示每组测试的运行时间,单位为 s。

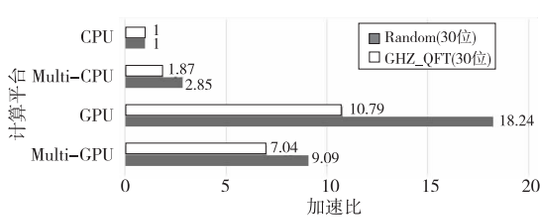


Figure 5　Comparison of performance speedup between multi-GPU version and other versions (30 bits)

图 5　多 GPU 版本与其它版本性能加速比比较(30 位)

图 5 中横轴表示每组测试的加速比(以单 CPU 性能为基准),纵轴表示每组测试所使用的计算平台。

通过数据对比可以发现,对于 2 个测试算例而言,多 GPU 版本的代码均可以获得明显的加速效果。对于 Random 算例,多 GPU 版本的运行时间为 34.1 s,相较于单 CPU 版本获得的加速比高达 9.09 倍,相较于多 CPU 版本获得的加速比大约为 3.19 倍;而对于 GHZ_QFT 算例,多 GPU 版本的运行时间为 32.5 s,相较于单 CPU 版本获得的加速比高达 7.04 倍,相较于多 CPU 版本获得的加速比大约为 3.76 倍。

但是,对比单 GPU 版本与多 GPU 版本的运行时间可以发现,单 GPU 版本的加速效果要优于多 GPU 版本的。这主要是因为多 GPU 版本在运行时,GPU 之间需要进行通信,从而产生了额外的开销,导致速度降低,这是不可避免的损失,而单 GPU 版本不涉及此类问题,所以加速效果更加明显。换句话说,在模拟的量子位较少,GPU 显存充足的情况下,还是首选单 GPU 版本来进行量子系统的模拟。同时,虽然多 CPU 版本也存在通信问题,但其计算收益远远大于通信带来的额外开销,所以相较于单 CPU 版本可以获得明显的性能提升。

30 个量子位的量子系统使用单 GPU 模拟需要消耗 16 GB 内存空间,多 GPU 版本的总内存消耗为 48 GB,即每个 GPU 消耗 12 GB。由于 $B1$ 与 $B2$ 缓冲区的实现,相较于单 GPU 版本,多 GPU 版本引入了 4 GB 的额外内存开销。

4.2　量子位扩展

本节设计了另一组测试实验,仍旧使用 Random 与 GHZ_QFT 2 个测试算例,将被模拟系统的量子位提高到 31 位,同样分别在单 CPU、多 CPU、单 GPU 和多 GPU 4 个平台上进行测试,测试结果分别如图 6 与图 7 所示。

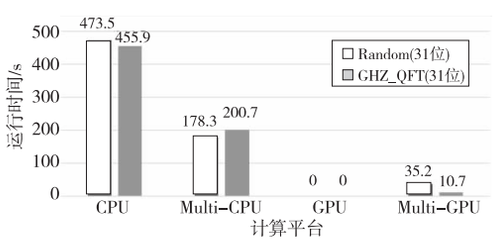


Figure 6　Comparison of simulation time between multi-GPU version and other versions (31 bits)

图 6　多 GPU 版本与其它版本模拟时间比较(31 位)

图6中横轴表示每组测试所使用的计算平台，纵轴表示每组测试的运行时间，单位为s。

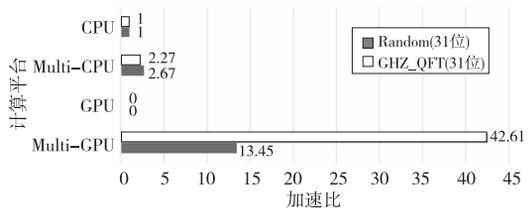


Figure 7 Comparison of performance speedup between multi-GPU version and other versions (31 bits)

图7 多GPU版本与其它版本性能加速比比较(31位)

图7中横轴表示每组测试的加速比(以单CPU性能为基准)，纵轴表示每组测试所使用的计算平台

通过数据对比可以发现，对于2个测试算例而言，多GPU版本的代码均可以获得明显的加速效果。对于Random算例，多GPU版本相较于单CPU版本获得的加速比高达13.45倍，相较于多CPU版本获得的加速比大约为5.04倍；而对于GHZ_QFT算例，多GPU版本的运行时间为10.7 s，相较于单CPU版本获得的加速比高达42.61倍，相较于多CPU版本获得的加速比大约为18.76倍。

然而，由于显存不足，对于这2个算例，单个同样型号的GPU均无法完成模拟测试，这说明改进后的多GPU版本代码不仅能够带来明显的性能加速，而且还确实解决了GPU显存不足的问题。此外，GHZ_QFT算例模拟的是更加规整的实际量子线路，相对于随机生成的Random算例，它取得的加速效果更加显著，也更加具有实际研究意义。

4.3 结论

通过上述实验，可以得出有关多GPU版本QuEST量子线路模拟器的3个结论：

(1)多GPU版本代码相对于CPU代码能够大大加速程序运行，加速效果显著；

(2)多GPU版本代码能够模拟具有更多量子位的量子系统；

(3)多GPU版本代码在通信较少的情况下，能够得到比单GPU更好的加速效果。

5 结束语

本文首先详细分析了QuEST量子线路模拟器的算法和可扩展性设计。QuEST目前提供单CPU、多CPU和单GPU 3个版本，虽然单GPU版本的QuEST模拟速度非常快，但是当模拟的位数增加到31位时，模拟所需要的内存已经超过目前市面上内存最大的GPU，所以单GPU版本最多只能模拟30个量子位的系统。当量子位数超过30位时，只能使用多CPU版本进行模拟。为了使QuEST能够模拟更多的位数同时速度尽可能地快，本文开发了多GPU版本，能够模拟超过30位的量子算法，解决了单GPU内存不足的问题。测试结果表明，与多CPU版本相比，本文开发的多GPU版本相较于单CPU版本能够获取7～9倍的性能提升，相较于多CPU版本可以获得3倍的性能提升。

未来将把工作重点放在优化GPU代码上，比如使用各种CUDA的优化技术来对QuEST程序进行加速，使其获得更好的加速效果，也会尝试使用RDMA技术优化结点间的通信，使其通信效率更高；同时，本文工作并没有修改QuEST模拟器的框架，但目前已经有一些工作利用张量网络和张量积解决量子线路模拟器内存消耗过大的问题[14-17]，如何使用这一方法优化QuEST将是接下来的一个研究方向。

参考文献：

[1] Construction and evaluation of gold standards for patent classification—A case study on quantum computing[EB/OL]. [2020-04-01]. https://www.sciencedirect.com/science/article/pii/S0172219019300791.

[2] Two-factor authentication using biometric based quantum operations[EB/OL]. [2020-04-01]. https://www.sciencedirect.com/science/artcle/pii/S0172219019300791.

[3] Recent advances in quantum machine learning[EB/OL]. [2020-02-20]. https://onlinelibrary.wiley.com/doi/pdf/10.1002/que2.34.

[4] Kandala A, Mezzacapo A, Temme K, et al. Hardware-efficient variational quantum eigensolver for small mole-cules and quantum magnets[J]. Nature, 2017,549(7671):242-246.

[5] ProjectQ: An open source software framework for quantum computing[EB/OL]. [2018-01-31]. https://quantum-journal.org/papers/q-2018-01-31-49/pdf.

[6] qHiPSTER: The quantum high performance software testing environment[EB/OL]. [2016-05-12]. https://arxiv.org/pdf/1601.07195.pdf.

[7] Distributed memory techniques for classical simulation of quantum circuits[EB/OL]. [2018-06-21]. https://arxiv.org/pdf/1801.01037.pdf.

[8] Jones T, Brown A, Bush I, et al. QUEST and high performance simulation of quantum computers[J]. Scientific Reports, 2019, 9(1): 1-11.

[9] Improved classical simulation of quantum circuits dominated by Clifford gates[EB/OL]. [2016-05-09]. https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.116.250501.

[10] Amariutei A, Caraiman S. Parallel quantum computer simulation on the GPU[C]// Proc of the 15th International Conference on System Theory, Control and Computing, 2011: 1-6.

[11] Zhang P, Yuan J, Lu X. Quantum computer simulation on multi-GPU incorporating data locality[C]// Proc of International Conference on Algorithms and Architectures for Parallel Processing, 2015: 241-256.

[12] Liu S, Li Y, Duan R. Distinguishing unitary gates on the IBM quantum processor[J]. Science China(Information Sciences), 2019, 62(7): 210-216.

[13] Gutiérrez E, Romero S, Trenas M A, et al. Quantum computer simulation using the CUDA programming model[J]. Computer Physics Communications, 2010, 181(2): 283-300.

[14] Arute F, Arya K, Babbush R, et al. Quantum supremacy using a programmable superconducting processor[J]. Nature, 2019, 574(7779): 505-510.

[15] Li K, Han M X, Qu D X, et al. Measuring holographic entanglement entropy on a quantum simulator[J]. npj Quantum Information, 2019, 5(1): 467-488.

[16] Exploiting the causal tensor network structure of quantum processes to efficiently simulate non-Markovian path integrals[EB/OL]. [2019-12-16]. https://arxiv.org/pdf/1902.00315.pdf.

[17] Quantum-teleportation-inspired algorithm for sampling large random quantum circuits[EB/OL]. [2020-02-26]. https://arxiv.org/pdf/1901.05003.pdf.

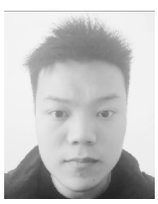
作者简介:

张亮(1997-),男,江苏连云港人,硕士生,CCF 会员(A0457G),研究方向为计算机体系结构。**E-mail**: 418761644@qq.com

ZHANG Liang, born in 1997, MS candidate, CCF member (A0457G), his research interest includes computer architecture.

常旭(1997-),男,辽宁锦州人,硕士生,CCF 会员(C3744G),研究方向为计算机体系结构。**E-mail**: xuchang15@nudt.edu.cn

CHANG Xu, born in 1997, MS candidate, CCF member(C3744G), his research interest includes computer architecture.



秦志凯(1998-),男,河南焦作人,硕士生,研究方向为计算机体系结构。**E-mail**: qinzhikai@nudt.edu.cn

QIN Zhi-kai, born in 1998, MS candidate, his research interest includes computer architecture.

沈立(1976-),男,四川成都人,博士,教授,CCF 会员(12903S),研究方向为多核/众核体系结构、运行时和编译优化、高性能计算。**E-mail**: lishen@nudt.edu.cn

SHEN Li, born in 1976, PhD, professor, CCF member(12903S), his research interests include multi/many-core architecture, runtime and compilation optimization, and high performance computing.