基于 CPU /GPU 异构平台的连续变量量子密钥分发多维数据协调

穆健健，郭大波*，马识途，贺超
山西大学物理电子工程学院，山西 太原 030006

摘要 针对当前连续变量量子密钥分发系统数据协调运算速率低的问题，采用中央处理器/图形处理器(CPU/GPU)异构平台实现了多维数据协调算法的并行加速运算，提出了对于异构计算要求的大规模校验矩阵静态双向十字链表及多维并行协调算法。在该平台上对码长为 2.048×10^5 的情况进行了仿真计算。通过仿真可获取收敛信噪比和协调计算时间，并计算得出协调速率、密钥传输距离和协调效率。结果表明：当码长为 2.048×10^5 时，在保证协调效率的前提下，采用 CPU/GPU 异构平台并行加速的协调速率为 CPU 平台的 5 倍。

关键词 量子光学；量子密钥分发；中央处理器/图形处理器异构平台；多维数据协调；低密度奇偶校验码；稀疏矩阵
中图分类号 O431.2 **文献标识码** A **doi**：10.3788/LOP56.152702

Multidimensional Data Reconciliation for Continuous-Variable Quantum Key Distribution Based on CPU/GPU Heterogeneous Platform

Mu Jianjian, Guo Dabo*, Ma Shitu, He Chao
College of Physics and Electronic Engineering, Shanxi University, Taiyuan, Shanxi 030006, *China*

Abstract Current continuous-variable quantum key distribution systems suffer from low computing speed for data reconciliation. Herein, we address this problem by implementing a parallel acceleration for a multidimensional data reconciliation algorithm based on a central processing unit/graphics processing unit (CPU/GPU) heterogeneous platform. To meet the special requirements of heterogeneous parallel computing, we propose a static two-way crosslinked list to store a hyperscale low-density parity-check matrix. We also propose a parallel reconciliation algorithm. A simulation experiment is carried out on the heterogenous platform with a code length of 2.048×10^5. Reconciliation speed, key transmission distance, and reconciliation efficiency are calculated based on the simulation results of the convergence signal-to-noise ratio and time of reconciliation. Results show that when the code length is 2.048×10^5, the reconciliation speed of parallel acceleration on the CPU/GPU heterogeneous platform is five times faster than that on the CPU platform.

Key words quantum optics; quantum key distribution; central processing unit/graphics processing unit heterogeneous platform; multidimensional data reconciliation; low density parity check code; sparse matrix
OCIS codes 270.5565; 270.5568; 270.5585

1 引 言

当今社会信息传输越来越频繁，如何保证信息的安全传输就显得尤为重要。作为量子密码中发展最为成熟的技术，量子密钥分发(QKD)具有理论上绝对安全性[1-2]。通过使用 QKD 以及后处理操作可以使合法的通信双方共享安全密钥，从而保证信息的安全传输。QKD 的发展是从第一个量子密钥分发协议开始的，将这一协议称为 BB84 协议。随后各国学者对其进行更深入的研究，提出了效率更

收稿日期：2019-02-19；**修回日期**：2019-02-28；**录用日期**：2019-03-05
基金项目：山西省基础研究项目(201801D121118)
* **E-mail**：dabo_guo@sxu.edu.cn

高和实用性更强的 QKD 协议，主要包括 B92 协议、PBC00 协议、与测量设备无关 QKD 协议（MDI-QKD）以及改进的 PBC00 协议[3]等。QKD 技术的研究主要分为两个方向：离散变量量子密钥分发（DVQKD）和连续变量量子密钥分发（CVQKD）。关于 CVQKD 的发展，2002 年由 Grosshans 等[4]提出了基于高斯调制相干态[5]和平衡零差探测器[6]的 GG02 协议。该协议的最后一步就是后处理操作，其关键部分为数据协调（即纠错编码）和密性放大[7]。通过数据协调，通信双方得到的密钥序列可以保证完全一致。再经过密性放大，虽然密钥减少，但是可以保证窃听者几乎无法获取密钥信息。

在实际应用中，针对 CVQKD 协议，国内外研究人员先后提出了不同的数据协调算法，例如，van Assche 等[8]提出的切片协调算法，该算法将发送方的连续变量序列 $\boldsymbol{X}$ 进行了量化，破坏了高斯对称性[9]，导致其在低信噪比下表现不佳，将安全传输距离限制在了 25 km；Silberhorn 等[10]提出符号协调算法[10]，在该算法中发送方揭示所发送的随机序列每个分量的绝对值，并将信息编码在每个分量的符号中，但是这种算法受到高斯调制的一些限制，由于高斯分布以 0 为中心，因此大多数数据具有小的绝对值，这就意味着区分其符号将变得非常困难。为了克服这个问题，采用后选择的方式[10-11]以舍弃小幅度数据并且仅保留更有意义的大幅度数据。然而，这种方法在安全性方面存在重大缺陷，可能允许窃听者实施一些强大的攻击。Leverrier 等[12]提出了一种不使用后选择方式的符号协调算法，即多维数据协调算法。该算法可通过两步定义码 $\boldsymbol{C}_X$ 来实现：1）在以 $\|\boldsymbol{X}\|$ 为半径的球体上定义一个 n 维汉明立方体 F_2^n 的同构图像 $\boldsymbol{Q}_X$；2）定义一个接近新信道容量的且可以有效译码的码，其 $\boldsymbol{C}_X \subset \boldsymbol{Q}_X$，并在实际操作中使用的是 LDPC 码。通过给定 $\boldsymbol{Q}_X$，把通信双方的信道转变为虚拟的二进制输入加性高斯白噪声信道，在长距离、低信噪比的情况下信道容量变化较小，可以保留信道的高斯对称性，而且通过旋转变换使得各个状态点之间的距离最大化，使得纠错码能在低信噪比下得到收敛，其安全传输距离超过了 50 km。

王云艳等[13]在中央处理器（CPU）平台采用多维协调算法，配合 LDPC 码，在 47 km 安全距离下实现了 8.61 kbit/s 的密钥传输率[14]。曾贵华团队在采用码长为 10^4 的码在图形处理器（GPU）上实现了多维数据协调，实验采用 25 MHz 的时钟频率[15]和 300 MHz 带宽的平衡零差探测器，译码速率达到了 25 Mbit/s[16]，最终达到的安全码率为52 kbit/s。Golub 等[17]对比了相同数据量的 LDPC 码在 CPU 和 GPU 平台上进行译码算法的译码速率，结果表明，GPU 平台的数据处理速率是 CPU 平台的 9 倍。

目前，CVQKD 主要受信道噪声和探测设备的影响，通信双方无法有效提取所有理论上可以获得的信息，限制了密钥的安全码率和传输距离。在数据协调方面，根据香农信息论，通过使用较长的码可以降低收敛信噪比，从而合理地接近信道的香农限。根据研究报道，码长在 2×10^5 附近时，CVQKD 数据协调在低信噪比实现收敛，而更长的码长对协调效率的提高增益不大，反而导致协调计算量显著增大。码长增加则会减缓计算速度、延长计算时间，降低 CVQKD 系统的实用性。

基于上述计算速度慢的问题，本文对基于 LDPC 的多维数据协调算法采用 CPU/GPU 异构平台进行并行化加速运算，将数据协调算法的并行译码用 CUDA（Compute Unified Device Architecture）编程模型实现，提高了协调速率，减少了时延。并且采用静态双向十字链表存储 LDPC 码的稀疏校验矩阵，大大减少了存储校验矩阵所用的空间，节约了读取校验矩阵所用的时间。

2 CVQKD 多维数据协调方案

多维协调算法[12]针对的是高斯调制的情况，采用具有合适信噪比的编码方案，实现了收敛信噪比的降低，从而延长了安全传输距离。编码策略为：给定 $\boldsymbol{y}\in S^{n-1}$（$S^{n-1}$ 表示球体），需要定义以 0 为中心且包含 $\boldsymbol{y}$ 的立方体 $\boldsymbol{Q}_y$，使得给定 $\boldsymbol{Q}_y$ 中 $\boldsymbol{y}$ 的先验分布是均匀的。描述 $\boldsymbol{Q}_y$ 可以通过描述转换规范立方体的正交变换来实现，Bob 随机选择规范立方体 D^n 的顶点 $\boldsymbol{U}$，$\boldsymbol{U}\in S^{n-1}$，再向 Alice 提供将 $\boldsymbol{y}$ 映射为 $\boldsymbol{U}$ 的正交变换 $\boldsymbol{M}(\boldsymbol{y},\boldsymbol{u})$，满足 $\boldsymbol{M}(\boldsymbol{y},\boldsymbol{u})\boldsymbol{y}=\boldsymbol{u}$，该变换定义了立方体 $\boldsymbol{Q}_y$。最后，Alice 可以根据 $\boldsymbol{x}$ 和 $\boldsymbol{M}(\boldsymbol{y},\boldsymbol{u})$ 恢复出 $\boldsymbol{U}$。

Alice 和 Bob 把 d 个连续变量构成 d 维向量，采取按照顺序直接组合的方法，可表示为

$$\begin{cases}\boldsymbol{Y}=\boldsymbol{X}+\boldsymbol{Z}\\ \boldsymbol{X}\sim N(0,\Sigma^2)d\\ \boldsymbol{Z}\sim N(0,\delta^2)d\end{cases},\tag{1}$$

式中：$\boldsymbol{X}$ 和 $\boldsymbol{Y}$ 分别为 Alice 和 Bob 拥有的 d 维向量；$\boldsymbol{Z}$ 为信道噪声；$N(\cdot)$ 表示正态分布；Σ^2 为 Alice 端的信号调制方差；δ^2 为信道的噪声方差。

2.1 连续高斯向量的球面化

Alice 和 Bob 分别将 $\boldsymbol{X}$ 和 $\boldsymbol{Y}$ 归一化，可表示为

$$\begin{cases}\boldsymbol{x}=\dfrac{\boldsymbol{X}}{\|\boldsymbol{X}\|}\\ \boldsymbol{y}=\dfrac{\boldsymbol{Y}}{\|\boldsymbol{Y}\|}\end{cases},\tag{2}$$

式中：$\|\boldsymbol{X}\|=\sqrt{\langle\boldsymbol{X},\boldsymbol{X}\rangle}$、$\|\boldsymbol{Y}\|=\sqrt{\langle\boldsymbol{Y},\boldsymbol{Y}\rangle}$，其中 $\langle\cdot\rangle$ 表示向量的点积运算。通过上述归一化过程，d 维状态点就从欧氏空间 D^d 映射到了单位球面空间 S^{d-1}。将变量空间从非均匀的高斯分布变为均匀高斯分布[18]。

2.2 在单位球面 S^{d-1} 上选择先验概率均匀分布的码字空间

Bob 在单位球面上随机选取 d 个距离最大的状态点构成 $\boldsymbol{u}$（其中包含旋转操作），可表示为

$$\boldsymbol{u}=\left[\frac{-1}{\sqrt{d}},\frac{1}{\sqrt{d}}\right]^d。\tag{3}$$

$\boldsymbol{u}$ 的具体选择过程为：先随机生成二进制串 $(b_1,b_2,\cdots,b_d)$，再进行操作

$$\boldsymbol{u}=\left[\frac{(-1)^{b_1}}{\sqrt{d}},\frac{(-1)^{b_2}}{\sqrt{d}},\cdots,\frac{(-1)^{b_d}}{\sqrt{d}}\right]。\tag{4}$$

接下来在 Bob 端计算旋转矩阵 $\boldsymbol{M}(\boldsymbol{y},\boldsymbol{u})$，其满足 $\boldsymbol{M}(\boldsymbol{y},\boldsymbol{u})\boldsymbol{y}=\boldsymbol{u}$，再通过经典认证信道将其发送给 Alice。

2.3 Alice 端进行旋转

Alice 端的旋转可表示为

$$\boldsymbol{M}(\boldsymbol{y},\boldsymbol{u})\boldsymbol{x}=\boldsymbol{v},\tag{5}$$

式中：$\boldsymbol{v}=\boldsymbol{u}+\boldsymbol{s}$，$\boldsymbol{s}$ 为 $\boldsymbol{u}$ 的误差。

2.4 旋转矩阵 $\boldsymbol{M}(\boldsymbol{y},\boldsymbol{u})$ 的计算

对于 $d=8$，存在由 $D^{8\times8}$ 空间的 8 个正交矩阵组成的族（不唯一）$(\boldsymbol{A}_1,\boldsymbol{A}_2,\cdots,\boldsymbol{A}_d)$，其中 $\boldsymbol{A}_1$ 是 8×8 的单位矩阵，而且对于 $i,j>1$，满足

$$\{\boldsymbol{A}_i,\boldsymbol{A}_j\}=-2\delta_{ij}\boldsymbol{A}_1,\tag{6}$$

式中：$\{\boldsymbol{A},\boldsymbol{B}\}=\boldsymbol{AB}+\boldsymbol{BA}$；$\delta_{ij}$ 为单位冲激函数；$(\boldsymbol{A}_1,\boldsymbol{A}_2,\cdots,\boldsymbol{A}_d)$ 在实际应用中是已经确定的。因此，旋转矩阵满足的公式为

$$\boldsymbol{M}(\boldsymbol{y},\boldsymbol{u})=\sum_{i=1}^{d}\alpha_i(\boldsymbol{y},\boldsymbol{u})\boldsymbol{A}_i,\tag{7}$$

式中：$[\alpha_1(y_1,u_1),\alpha_2(y_2,u_2),\cdots,\alpha_d(y_d,u_d)]$ 是 $\boldsymbol{u}$ 在正交基 $[\boldsymbol{A}_1y_1,\boldsymbol{A}_2y_2,\cdots,\boldsymbol{A}_dy_d]$ 上的坐标，其中 y_d 表示多维空间的坐标轴。所以 $\boldsymbol{M}(\boldsymbol{y},\boldsymbol{u})$ 的计算可转化为 $[\alpha_1(y_1,u_1),\alpha_2(y_2,u_2),\cdots,\alpha_d(y_d,u_d)]$ 的计算。又因为 $[\boldsymbol{A}_1y_1,\boldsymbol{A}_2y_2,\cdots,\boldsymbol{A}_dy_d]$ 为可逆矩阵，所以可表示为

$$[\alpha_1(y_1,u_1),\alpha_2(y_2,u_2),\cdots,\alpha_d(y_d,u_d)]^{\mathrm{T}}=[\boldsymbol{A}_1y_1,\boldsymbol{A}_2y_2,\cdots,\boldsymbol{A}_dy_d]^{-1}\boldsymbol{u}。\tag{8}$$

2.5 数据协调的安全密钥量估计

采用逆向协调的方式进行数据协调[19]，讨论集体攻击下的安全密钥量估计，所以发送方 Alice 与接收方 Bob 之间的互信息 I_{AB} 和 Bob 可能泄露给 Eve 的信息量 χ_{BE} 的大小关系决定了双方能否安全通信。在实际协调过程中，安全密钥速率可表示为

$$T=\beta I_{\mathrm{AB}}-\chi_{\mathrm{BE}},\tag{9}$$

式中：β 为实际可获取的信息与理论上可获得的信息的比值，即协调效率。$T>0$ 时，表示通信是安全的。因此，在实际操作中要保证效率尽可能大，而且要想提取到安全密钥，需保证协调效率>90% [13]。β 的计算式[20]为

$$\beta=\frac{R}{\frac{1}{2}\log_2(1+R_{\mathrm{SN}})},\tag{10}$$

式中：R 为码率；R_{SN} 为收敛信噪比。

基于高斯调制相干态和零差探测的 CVQKD 协议，Alice 和 Bob 之间的总噪声 χ_{tot} 可表示为

$$\chi_{\mathrm{tot}}=\chi_{\mathrm{line}}+\chi_{\mathrm{hom}}/T,\tag{11}$$

$$\chi_{\mathrm{line}}=1/T-1+\varepsilon,\tag{12}$$

$$\chi_{\mathrm{hom}}=(1+v_{\mathrm{el}})/\eta-1,\tag{13}$$

式中：χ_{line} 为量子信道噪声；T 为信道的透射率；χ_{hom} 为 Bob 端使用零拍探测器引入的噪声；ε 为传输过程中存在的额外噪声；v_{el} 为探测器电子设备引入的噪声；η 为探测器的效率。发送端 Alice 将真空态沿 x_{A} 和 p_{A} 方向均进行平移得到相干态，其中 x_{A} 为振幅分量，p_{A} 为相位分量，两者都服从均值为 0，方差为 $V_{\mathrm{A}}N_0$ 的高斯分布，而 N_0 为真空噪声功率，V_{A} 为发送端 Alice 的信号功率，上述的噪声均是以 N_0 为单位。I_{AB} 可计算为

$$I_{\mathrm{AB}}=\frac{1}{2}\log_2\frac{V_{\mathrm{B}}}{V_{\mathrm{B|A}}}=\frac{\eta T(V+\chi_{\mathrm{tot}})}{\eta T(1+\chi_{\mathrm{tot}})}=\frac{1}{2}\log_2\frac{V+\chi_{\mathrm{tot}}}{1+\chi_{\mathrm{tot}}},\tag{14}$$

式中：V_{B} 为 Bob 端的功率值；$V_{\mathrm{B|A}}$ 为条件方差，表示已知 Alice 功率的条件下 Bob 端的功率；$V=V_{\mathrm{A}}+1$。取 $V_{\mathrm{A}}=18.5$、$T=0.151$、$\varepsilon=0.01$、$\eta=0.606$、$v_{\mathrm{el}}=0.041$，利用(14)式计算出 $I_{\mathrm{AB}}\approx244.05$ kbit/s。χ_{BE} 为 Eve 从 Bob 端获取的信息量的 Holevo 界[21]，可通过文献[13]给出的 Holevo 限计算公式得出 $\chi_{\mathrm{BE}}=222.97$ kbit/s。多维协调算法的整体方案如图 1 所示。

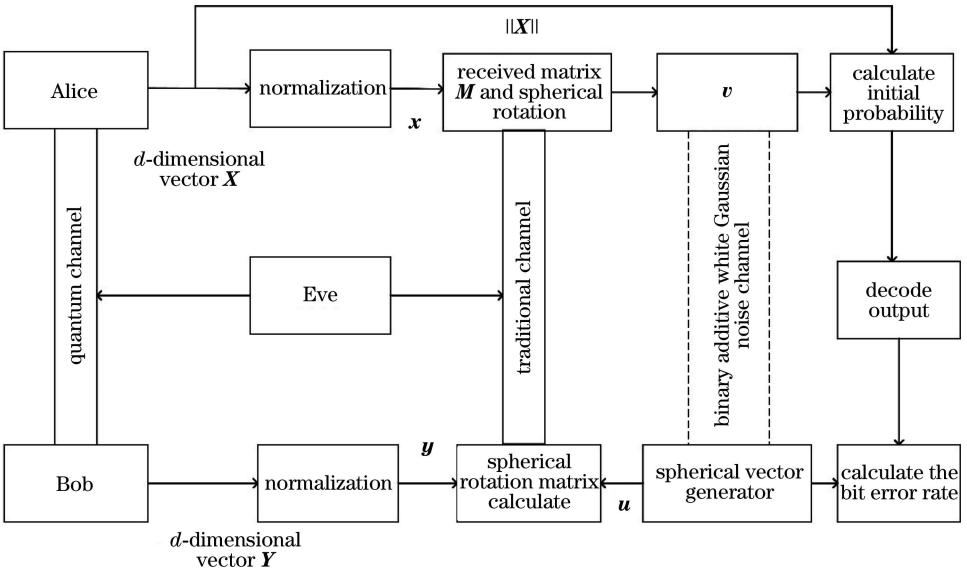

图 1 多维协调算法整体示意图

Fig. 1 Overall diagram of multidimensional reconciliation algorithm

3 基于二进制 LDPC 码的译码算法

3.1 计算初始概率

使用二进制 LDPC 码进行译码时，需要计算信道传递给变量节点的初始概率，即计算向量 $\boldsymbol{v}$ 确定的情况下，向量 $\boldsymbol{u}$ 中的二进制串对应位置是 0 或 1 的概率。映射到球面空间就是计算 $\boldsymbol{u}$ 中对应位为 $\frac{1}{\sqrt{d}}$或$\frac{-1}{\sqrt{d}}$的概率。计算初始概率，文献[18]给出的计算公式为

$$P_i(s)=P_r[u_i=u(s)\mid v_i]=K\exp\left\{\frac{-[\|\boldsymbol{Y}\|v_i-\|\boldsymbol{X}\|u(s)]^2}{2\sigma^2}\right\}\Big/(2\pi\sigma^2)^{\frac{1}{2}},\tag{15}$$

式中：P_i 为计算得到的概率；P_r 为后验概率；u_i 和 v_i 分别为向量 $\boldsymbol{u}$ 和 $\boldsymbol{v}$ 的第 i 个分量；$s\in(b_1,b_2,\cdots,b_d)$为二进制串中的码字；σ^2 为高斯分布的方差；$u(s)=\frac{(-1)^s}{\sqrt{d}}$ 为 s 确定后的球面向量 $\boldsymbol{u}$ 中的元素；K 为归一化因子，使得 $P_i(0)+P_i(1)=1$。

3.2 对数似然比置信传播译码过程

对数似然比置信传播(LLR BP)译码过程中采用似然比来表示概率消息，以加法运算代替了原来的乘法运算，降低了计算复杂度，减少了运算时间。因此，结合上述的协调过程介绍译码的具体实现[22]：f_{mn} 是从校验节点 m 传递给变量节点 n 的外部概率信息，q_{nm} 是从变量节点 n 传递给校验节点 m 的外部概率信息，F_m 是与校验节点 m 相连的所有变量节点的集合。$F_{m\backslash n}$是与校验节点 m 相连的变量节点的集合，不含变量节点 n，Q_n 是与变量节点 n 相连的所有校验节点的集合，$Q_{n\backslash m}$是与变量节点 n 相连的校验节点的集合，不含校验节点 m，q_n 表示某一变量节点。

LLR BP 算法如下（迭代次数用信息符号的上标来表示）。

1) 信道初始化

初始概率似然比可表示为

$$L^{(0)}(q_{nm})=L(P_i)=\log_2\frac{P_i(0)}{P_i(1)}=\log_2\frac{\left\{u_i=\frac{1}{\sqrt{d}}\mid v_i\right\}}{\left\{u_i=\frac{-1}{\sqrt{d}}\mid v_i\right\}},\tag{16}$$

式中：$L^{(0)}$ 为信道传递给变量节点的初始概率似然比。

2) 迭代处理

①校验节点消息处理

在第 l 次迭代中，校验节点信息的计算公式为

$$\tanh\left[\frac{1}{2}L^{(l)}(f_{mn})\right]=\prod_{n'\in(F_m\backslash n)}\tanh\left[\frac{1}{2}L^{(l-1)}(q_{n'm})\right],\tag{17}$$

式中：$L^{(l)}$ 为第 l 次迭代时的似然比，l 表示迭代次数；$q_{n'm}$为变量节点 n'传递给校验节点 m 的外部概率信息。

②变量节点消息处理

在第 l 次迭代中，变量节点信息的计算公式为

$$L^{(l)}(q_{nm})=L(P_i)+\sum_{m'\in(Q_n\backslash m)}L^{(l)}(f_{m'n}),\quad(18)$$

式中：m'为校验节点，范围是$Q_{n\backslash m}$。

3）译码处理

所有变量节点判决信息的计算公式为

$$L^{(l)}(q_n)=L(P_i)+\sum_{m\in Q_n}L^{(l)}(f_{mn})。\quad(19)$$

对所有变量节点信息进行硬判决：当$L^{(l)}(q_n)<0$时，译出的码字向量元素为1；否则，译出的码字向量元素为0。

4）控制迭代次数

当满足$\boldsymbol{H}\boldsymbol{g}^{\mathrm{T}}=0$（其中$\boldsymbol{H}$为LDPC码的校验矩阵；T为转置；$\boldsymbol{g}$为译出的码字向量）或达到最大迭代次数时，译码结束。如果不满足，则返回步骤2）。

4 基于CPU/GPU异构平台的多维数据协调实现

4.1 CUDA编程模型介绍

由于GPU的多线程结构设计非常适合用于并行计算，2007年NVIDIA公司基于本公司生产的GPU推出了CUDA编程模型[23]。该模型引入了主机端和设备端的概念用来区分CPU和GPU。一个完整的CUDA程序包含主机函数和内核函数，其中主机函数在CPU上串行执行，即一般的C/C++代码，而内核函数在GPU上并行执行。主机函数只能被CPU调用，而内核函数可以被CPU和GPU调用。CPU上执行的主机函数负责内核函数启动之前的设备初始化，以及数据准备和内核函数之间的一些串行运算。GPU上执行的内核函数在线程格和线程块两个层次中并行完成。

CUDA程序的执行结构：CUDA程序执行过程中主机函数和内核函数交替运行，CPU和GPU快速、高效、协同地完成高性能计算任务。程序从CPU端的主机函数开始运行，当运行到内核函数时，切换到GPU端，启动多个GPU线程来并行执行内核函数。每个线程块均包含多个线程，这些线程对不同的数据执行相同的指令，实现块内线程并行，同时每个线程格内的不同块之间实现块并行。当GPU端完成运算后，返回主机函数，CPU继续执行串行代码，直到程序运行结束。CUDA程序执行结构如图2所示。

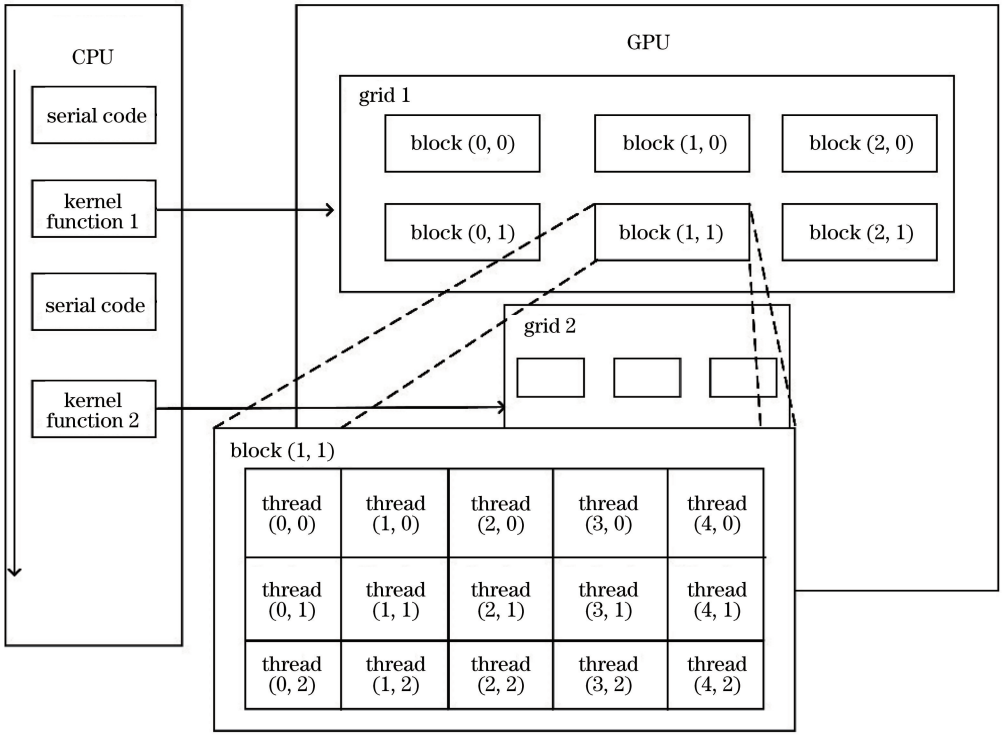

图2 CUDA程序的执行过程图

Fig. 2 Diagram of CUDA program execution process

4.2 LDPC码的校验矩阵存储

根据信息论的研究，稀疏校验矩阵的尺寸越大，其中1的分布就越随机，这样构造的LDPC码的纠错性能也就越好。所以本研究采用N为10^5量级的校验矩阵（N表示校验矩阵的列数）。在GPU上进行译码时，需要使用校验矩阵$\boldsymbol{H}$中的数据，由于

$\boldsymbol{H}$ 矩阵尺寸大，且其中不同行和列上 1 的位置和个数不同，以及 CUDA 核函数不支持使用三级指针作为参数，所以采用静态双向十字链表示存储 $\boldsymbol{H}$ 矩阵，且只存储 $\boldsymbol{H}$ 矩阵中 1 的位置。具体过程如下：1）在 CPU 端获取到提供的稀疏校验矩阵的行数 n_{rows}、列数 n_{cols}、1 的个数 N_{um}，以及元素 1 的行和列坐标信息；2）申请大小为 $N_{\text{um}}+n_{\text{rows}}+n_{\text{cols}}$ 的静态连续内存，将所有的非零元素信息存储在这片区域中。该区域中存储的稀疏矩阵非 0 元素的节点是以结构体数组的形式定义的。结构体的成员中 row 和 col 分别表示节点的行和列坐标，pr 与 lr 分别表示变量节点与校验节点更新的似然比信息，left 表示节点所在行的上一个节点的列信息；right 表示节点所在行的下一个节点的列信息；up 表示节点所在列的上一个节点的行信息；down 表示节点所在列的下一个节点的行信息。在静态链表中，这些位置信息可被定义为 int 类型，而不是原来的指针类型，避免了使用三级指针。静态双向十字链表的数据结构如下：

```
typedef struct
{
  int row,col;
  int left,right,up,down;
  double pr,lr;
}nodentry;
typedef struct
{
  int n_rows,n_cols;
  int Num;
  nodentry *rows; //指向存储列信息的空间
  nodentry *cols;//指向存储行信息的空间
  nodentry *e;//指向某个节点
  nodentry *mem;//指向存储非 0 元素的空间
}sparseMatr;
```

上述结构中的 rows 和 cols 表示两个链域，即行链域和列链域[16]。在行链域上 rows[m].left 存储的是前一个节点的列信息，rows[m].right 存储的是后一个节点的列信息。在列链域上，cols[n].up 存储上一个节点的行信息，cols[n].down 存储的是下一个节点的行信息。采用这样的十字双向链表不仅可以减少内存占用，而且可以很快定位非 0 元素，从而缩短译码时间。

4.3 多维数据协调的并行化实现

采用 CUDA 编程模型将多维数据协调过程分为主机执行和设备执行两部分。多维数据协调过程中的译码过程需要多次迭代才能收敛，而且每次迭代过程中存在大量的行和列运算，假设采用的 LDPC 码的码长为 N，其校验矩阵 $\boldsymbol{H}$ 的维数为 $M\times N$，对应有 M 个校验节点和 N 个变量节点。根据(17)式共有 M 个校验方程进行行运算；根据(18)式共有 N 个码字进行列运算。在每次迭代过程中，信息在校验节点和变量节点之间传递，更新校验节点和变量节点的似然比信息，则信息的传递过程互不影响。由于采用了 $\boldsymbol{H}$ 矩阵，M 和 N 对应的均是 10^5 量级，所以存在大量相同操作的重复运算，非常符合单指令多数据的并行计算模式[23]。因此，将译码的这部分分配到设备端进行处理。将协调算法的其余部分分配到主机端进行处理。整体执行过程如下。

1）主机程序读取 LDPC 码的稀疏校验矩阵 $\boldsymbol{H}$，将其用静态双向十字链表的方式进行存储；控制 Alice 端发送连续变量 $\boldsymbol{X}$ 给 Bob，并在 Bob 接收到连续变量 $\boldsymbol{Y}$ 后，在主机端对 $\boldsymbol{X}$ 和 $\boldsymbol{Y}$ 进行归一化，分别得到 $\boldsymbol{x}$ 和 $\boldsymbol{y}$，在球面上选择先验概率均匀分布的码字空间得到 $\boldsymbol{u}$。利用 $\boldsymbol{u}$ 和 $\boldsymbol{y}$ 得到旋转矩阵 $\boldsymbol{M}$；再由 $\boldsymbol{M}$ 和 $\boldsymbol{x}$ 计算得到 $\boldsymbol{v}$。通过计算得到校验子和似然比信息；调用 cudaMalloc 函数为译码函数的输入输出对象分配设备内存。其中，输入对象包括稀疏校验矩阵 $\boldsymbol{H}$，用于初始化变量节点的似然比信息、校验子信息和迭代信息，输出对象包括译出的码字。

2）主机端调用 cudaMemcpy 函数将主机端内存上的输入对象数据拷贝到 GPU 的设备内存上。其中，cudaMemcpy 函数的第 4 个参数设置为 cudaMemcpyHostToDevice。

3）主机函数调用内核函数。在设备端启动了 3 个内核函数，功能依次是变量节点初始化、更新校验节点信息和更新变量节点信息，这一部分在 GPU 上进行。

变量节点初始化。启动 N 个线程分别对应 $\boldsymbol{H}$ 矩阵的 N 列数据单元，调用内核函数的语法为：内核函数名<<<网格的维度，线程块的维度>>>(内核函数的输入输出参数)；将每个线程块内的线程数量设置为 1024，所以分配的线程块数量是 ceil(static_cast<float>(N)/1024.0f)。每个块内的线程数量最好设置为线程束 32 的倍数，而且不能超过设备关于每个线程块最大启动的线程数量限制。启动的每个线程都会通过一个 threadIdx 来标识，其不是一个常量而是索引[24]，在执行过程中会遍历 0～$N-1$。

校验节点信息更新：启动 M 个线程对应 $\boldsymbol{H}$ 的 M 行数据单元，按(17)式进行校验节点信息更新。变量节点信息更新：启动 N 个线程对应 $\boldsymbol{H}$ 的 N 列数据单元，按(18)式进行变量节点信息更新。

内核函数执行过程中将输入参数的值更新到每个节点，这样下一个内核函数就可以使用已经更新的值进行计算，通过这样的信息传递，纠错码逐渐收敛，当满足译码终止条件时，译码结束返回迭代次数。

4)主机函数调用 cudaMemcpy 将设备内存上译出的码字拷贝回主机内存，计算误码率。

5 仿真结果与分析

采用的硬件平台是 CPU 为 Intel Corei7-7700k 4.2 GHz，主机内存为 16 GB，GPU 设备为 NVIDIA Tesla K40C，CUDA 计算能力 3.5，CUDA 核个数 2880，设备内存为11421 MB。所用仿真软件为 Visual Studio2015 搭配 CUDA9.0。所用校验矩阵由 Mackay 随机构造法生成，协调算法为八维数据协调算法。选择码长为 2.048×10^5，迭代次数为 100，共 10 个分组进行统计。表 1 对比了 CPU 平台和 CPU/GPU 异构平台在不同码率时的数据协调结果。

表 1 码率不同时 CPU 和 CPU/GPU 异构平台的实验结果

Table 1 Experimental results of CPU platform and CPU/GPU heterogeneous platform at different code rates

R_{ate}	CPU			CPU/GPU		
	R_{SN}/dB	$A_{\mathrm{ve\text{-}time}}$/s	S_{peed}/(kbit·s^{-1})	R_{SN}/dB	$A_{\mathrm{ve\text{-}time}}$/s	S_{peed}/(kbit·s^{-1})
0.20	0.40	6.7027	30.55	0.45	1.2017	170.42
0.30	0.55	6.6238	30.92	0.55	1.2173	168.24
0.35	0.70	6.6042	31.01	0.70	1.2048	169.99
0.40	0.80	6.6016	31.02	0.90	1.2174	168.23
0.45	1.00	6.5714	31.16	1.10	1.1826	173.18
0.50	1.20	6.5535	31.25	1.20	1.2189	168.02

表 1 中 R_{ate} 为码率；R_{SN} 为收敛信噪比；$A_{\mathrm{ve\text{-}time}}$ 为协调的平均时间；S_{peed} 为协调速率，由码长除以协调的平均时间得出。由表 1 可以看出，在码率相同的情况下，CPU/GPU 异构平台的协调速率是 CPU 平台的 5 倍。这是因为异构平台在进行译码时，将数据复制到 GPU 上，在进行变量节点初始化时，会启动 2.048×10^5 个线程分别对应校验矩阵 $\boldsymbol{H}$ 的每一列，从上到下对变量节点初始化，所有线程都是同时进行的。而在 CPU 上执行时只会启动一个线程从第一列开始，从上到下执行完后，再执行第二列。校验节点信息更新和变量节点信息更新执行原理与此类似，这样做节省了译码时间。当码率从 0.2 增加到0.5时，两平台收敛信噪比均有所增加，这是因为码率增加导致 M 减少，也就是码所携带的校验信息减少(由码率计算公式 $R=1-\frac{M}{N}$ 可以看出)，在迭代译码时一些置信信息错误没有及时得到修正，导致收敛性能下降。当码率增加时，CPU 平台协调的平均时间有所减少但是变化不大，同样是因为校验次数减少引起的。异构平台协调的平均时间变化不大，这是因为在译码时该设备执行并行计算能力强，而且总的协调时间受码长和迭代次数影响较大，校验次数变化对总的时间的变化影响很小。在码长相同的情况下，表 2 计算了不同码率时两个平台的安全密钥量。

表 2 码率不同时 CPU 平台和 CPU/GPU 平台的密钥量结果

Table 2 Experimental results about rate-bit of CPU platform and CPU/GPU platform at different code rates

R_{ate}	CPU				CPU /GPU			
	R_{SN}/dB	β /%	$R_{\mathrm{ate\text{-}bit}}$/(kbit·s^{-1})	D_{istance}/km	R_{SN}/dB	β /%	$R_{\mathrm{ate\text{-}bit}}$/(kbit·s^{-1})	D_{istance}/km
0.20	0.40	82.40	−21.83	48.00	0.45	74.62	−54.81	47.75
0.30	0.55	94.89	8.61	47.25	0.55	94.89	8.61	47.25
0.35	0.70	91.44	0.19	47.50	0.70	91.44	0.19	47.50
0.40	0.80	94.34	7.26	47.00	0.90	86.39	−11.16	45.50
0.45	1.00	90.00	−3.32	45.00	1.10	84.08	−17.77	44.50
0.50	1.20	87.91	−7.44	44.00	1.20	87.91	−7.44	44.00

表 2 中 β 为协调效率；$R_{\mathrm{ate\text{-}bit}}$ 为安全密钥速率；D_{istance} 为传输距离。协调效率通过(10)式计算得出，安全密钥速率通过(9)式计算得出。仿真中采用 1550 nm 的单模光纤，损耗为 0.2 dB/km，传输距离由 $D_{\mathrm{istance}}=\frac{10-R_{\mathrm{SN}}}{0.2}$ 计算得出。由表 2 可以看出，有

的码对应的安全密钥率为负值，表示此次密钥分发失败，不能得到安全密钥。只有码率为 0.3 和 0.35 时，协调效率＞90%，得到了安全密钥。只有在保证能够获取到安全密钥的前提下，协调速率的提高才有意义。采用 CPU/GPU 方案后，部分码的协调效率变小，这是由于 GPU 平台的 CUDA C 语言的精度问题引起的实验结果的偏差，需要进一步优化程序消除影响。

6 结　　论

对基于高斯调制相干态的 CV-QKD 采用八维数据协调算法在 CPU/GPU 异构平台进行仿真，采用 LDPC 码作为纠错码，并用静态双向十字链表存储其校验矩阵。当连续变量长度为 2.048×10^5 时，在保证协调效率的前提下，采用 CPU/GPU 异构平台并行加速的协调速率为 CPU 平台的 5 倍，减少了时延，提高了 CV-QKD 的实用性。当码率为 0.5 时，同样采用八维协调算法的文献[13]报道的 CPU 平台协调速率为14.48 kbit/s(未报道更低码率的协调速率)；本文得到的 CPU 平台协调速率为 31.25 kbit/s，CPU/GPU 平台协调速率为 168.24 kbit/s，除去 CPU 计算能力造成的差异，协调速率是文献[13]结果的 9.4 倍。

需要改善的是寻找不规则 LDPC 码的最佳度分布，构造出高性能的码，以降低码的收敛信噪比，实现协调效率的提高，以及通过 GPU 端的优化进一步提高协调速率。

参 考 文 献

[1] Bennett C H, Brassard G. Quantum cryptography: public key distribution and coin tossing [J]. Theoretical Computer Science, 2014, 560: 7-11.

[2] Guo H, Li Z Y, Peng X. Quantum cryptography [M]. Beijing: National Defense Industry Press, 2016.
郭弘, 李政宇, 彭翔. 量子密码[M]. 北京: 国防工业出版社, 2016.

[3] Wang Z, Yao Z H, Gou L D, *et al*. Security analysis of three-state quantum key distribution protocol[J]. Laser & Optoelectronics Progress, 2017, 54(12): 122702.
王者, 姚治海, 苟立丹, 等. 三量子态量子密钥分发协议安全性分析[J]. 激光与光电子学进展, 2017, 54(12): 122702.

[4] Grosshans F, van Assche G, Wenger J, *et al*. Quantum key distribution using Gaussian-modulated coherent states[J]. Nature, 2003, 421(6920): 238-241.

[5] Grosshans F, Grangier P. Continuous variable quantum cryptography using coherent states [J]. Physical Review Letters, 2002, 88(5): 057902.

[6] Ma L X, Qin J L, Yan Z H, *et al*. Fast response balanced homodyne detector for continuous-variable quantum memory[J]. Acta Optica Sinica, 2018, 38(2): 0227001.
马丽霞, 秦际良, 闫智辉, 等. 用于连续变量量子存储的快速响应平衡零拍探测器[J]. 光学学报, 2018, 38(2): 0227001.

[7] Liu Y P, Guo J S, Cui J Y. Scheme design of highly efficient privacy amplification with fewer random seeds in quantum key distribution[J]. Acta Optica Sinica, 2017, 37(2): 0227002.
刘翼鹏, 郭建胜, 崔竞一. 高效短种子量子密钥分配保密放大方案设计[J]. 光学学报, 2017, 37(2): 0227002.

[8] van Assche G, Cardinal J, Cerf N J. Reconciliation of a quantum-distributed Gaussian key [J]. IEEE Transactions on Information Theory, 2004, 50(2): 394-400.

[9] Bloch M, Thangaraj A, McLaughlin S W, *et al*. LDPC-based Gaussian key reconciliation [C]//2006 IEEE Information Theory Workshop, 13-17 March 2006, Punta del Este, Uruguay. New York: IEEE, 2006: 116-120.

[10] Silberhorn C, Ralph T C, Lütkenhaus N, *et al*. Continuous variable quantum cryptography: beating the 3 dB loss limit [J]. Physical Review Letters, 2002, 89(16): 167901.

[11] Namiki R, Hirano T. Practical limitation for continuous-variable quantum cryptography using coherent states[J]. Physical Review Letters, 2004, 92(11): 117901.

[12] Leverrier A, Alléaume R, Boutros J, *et al*. Multidimensional reconciliation for a continuous-variable quantum key distribution [J]. Physical Review A, 2008, 77(4): 042325.

[13] Wang Y Y, Guo D B, Zhang Y H, *et al*. Algorithm of nultidimensional reconciliation for continuous-variable quantum key distribution[J]. Acta Optica Sinica, 2014, 34(8): 0827002.
王云艳, 郭大波, 张彦煌, 等. 连续变量量子密钥分发多维数据协调算法[J]. 光学学报, 2014, 34(8): 0827002.

[14] Dou L. Optimizing multidimensional reconciliation algorithm for continuous-variable quantum key distribution[J]. Acta Optica Sinica, 2016, 36(9): 0927001.
窦磊. 连续变量量子密钥分发多维数据协调算法优化[J]. 光学学报, 2016, 36(9): 0927001.

[15] Wang C, Huang D, Huang P, *et al*. 25 MHz clock continuous-variable quantum key distribution system over 50 km fiber channel[J]. Scientific Reports, 2015, 5: 14607.

[16] Liu S T, Wang X K, Guo D B. Accelerated computational implementation of reconciliation for continuous variable quantum key distribution on GPU[J]. Journal on Communications, 2017, 38(11): 2017222.
刘绍婷, 王晓凯, 郭大波. 连续变量量子密钥分发数据协调加速运算的 GPU 实现[J]. 通信学报, 2017, 38(11): 2017222.

[17] Golub G H, Van L Charles F. Matrix computations[M]. Yuan Y X, Transl. Beijing: Science Press, 2009.
Golub G H, Van L Charles F. 矩阵计算[M]. 袁亚湘, 译. 北京: 科学出版社, 2009.

[18] Lin Y, He G Q, Zeng G H. The application of LDPC codes in the multidimensional reconciliation of quantum key distribution[J]. Acta Sinica Quantum Optica, 2013, 19(2): 116-121.
林毅, 何广强, 曾贵华. LDPC 码在量子密钥分配多维协商算法中的应用[J]. 量子光学学报, 2013, 19(2): 116-121.

[19] Lu Z X, Yu L, Li K, *et al*. Reverse reconciliation for continuous variable quantum key distribution[J]. Scientia Sinica(Physica, Mechanica & Astronomica), 2009, 39(11): 1606-1612.
逯志欣, 于丽, 李康, 等. 基于逆向协调的连续变量量子密钥分发数据协调[J]. 中国科学: 物理学 力学 天文学, 2009, 39(11): 1606-1612.

[20] Jouguet P, Kunz-Jacques S, Leverrier A. Long-distance continuous-variable quantum key distribution with a Gaussian modulation[J]. Physical Review A, 2011, 84(6): 062317.

[21] He Z L, Guo D B, Wang X K. Security capacity of compound wiretap channel [J]. Laser & Optoelectronics Progress, 2015, 52(11): 112701.
贺转玲, 郭大波, 王晓凯. 复合窃听信道的安全容量[J]. 激光与光电子学进展, 2015, 52(11): 112701.

[22] Yuan D F, Zhang H G. The theory and application of LDPC code[M]. Beijing: Posts & Telecom Press, 2008: 75-79.
袁东风, 张海刚. LDPC 码理论与应用[M]. 北京: 人民邮电出版社, 2008: 75-79.

[23] Qiu D Y. GPGPU programming technology: from GLSL, CUDA to OpenCL [M]. Beijing: China Machine Press, 2011
仇德元. GPGPU 编程技术: 从 GLSL、CUDA 到 OpenCL[M]. 北京: 机械工业出版社, 2011.

[24] Kirk D B, Hwu W H. Programming massively parallel processors: a hands-on approach[M]. Zhao K Y, Wang Z H, Cheng Y C, Transl. Beijing: Tsinghua University Press, 2013
David B. Krik, Wen-mei W. Hwu .大规模并行处理器编程实战[M]. 赵开勇, 汪朝辉, 程亦超, 译. 北京: 清华大学出版社, 2013.