doi:10.11959/j.issn.1000-436x.2017222

连续变量量子密钥分发数据协调加速运算的 GPU 实现

刘绍婷，王晓凯，郭大波
（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度低等问题，用 GPU 实现了基于 LDPC 的 SEC 协议下数据协调算法的并行化加速运算，提出用静态双向十字链表的方法高效存储大规模稀疏校验矩阵，从而保证在不牺牲协调效率的前提下提高了译码速率。仿真实验结果表明，在信道信噪比为 4.9 dB 以上、2×10^5 个连续变量序列可靠协调以及协调效率为 91.71%的情况下，基于 Geforce GT 650 MB 的 GPU 和 2.5 GHz、8 GB 内存的 CPU 硬件平台，译码速率可达 16.4 kbit/s，相对于仅 CPU 平台，计算速度提高 15 倍以上。

关键词：连续变量量子密钥分发；数据协调；低密度奇偶码；静态链表；GPU 译码
中图分类号：TN918.91 **文献标识码**：A

Accelerated computational implementation of reconciliation for continuous variable quantum key distribution on GPU

LIU Shao-ting, WANG Xiao-kai, GUO Da-bo
(College of Physics and Electronic Engineering, Shanxi University,Taiyuan 030006, China)

Abstract: For the low computing speed of reconciliation for current continuous variable quantum key distribution，CPU&GPU-parallel reconciliation algorithms was designed based on LDPC of SEC protocol to speed up decoding computing. In order to raise decoding speed without sacrifice reconciliation efficiency, a static two-way cross linked list to efficiently store large scale sparse parity matrix was employed. The simulation experimental results show that the speed of the decoding rate reaches 16.4 kbit/s when the channel SNR is over 4.9 dB and the reliability of the 2×10^5 continuous variable quantum sequence, with reconciliation efficiency of 91.71%. The experimental based on the Geforce GT 650 MB GPU and the 2.5 GHz and 8 GB memory CPU hardware platform. Relative to the only CPU platform, computing speed increased by more than 15 times.

Key words: continuous variable quantum key distribution, reconciliation, low density parity check code, static linked list, GPU decoding

1 引言

随着计算机信息技术的迅速发展，信息安全占据着十分重要的地位。量子密钥分发（QKD）具有物理的无条件安全性，从而量子保密通信获得广泛关注。其中，连续变量量子密钥分发（CV-QKD）是量子保密通信领域中的一个重要分支，成为众多学者们研究的热点。

数据协调是量子密钥分发中的重要步骤，在量子信道上通信双方完成数据传输后，需要通过一个数据协调协议来纠正窃听或信道噪声所引入的误码，在本质上是一个纠错过程。数据协调被看作是一个非对称分布式信源编码过程[1]，通过对其中一个信源序列进行压缩，结合另一信源的全部信息进

收稿日期：2017-01-20；**修回日期**：2017-07-04
基金项目：山西省国际科技合作计划基金资助项目（No.2014081027-1）；山西省基础研究基金资助项目（No.2014011007-2）；山西省回国留学人员科研基金资助项目（No.2014-012）
Foundation Items: International Technology Cooperation Program of Shanxi Province (No.2014081027-1), The Basic Research Program of Shanxi Province (No.2014011007-2), Research Project Supported by Shanxi Scholarship Council of China (No.2014-012)

行联合译码，从而得到一条公共序列。

1993 年，Brassard 等 [2]第一次提出了数据协调（data reconciliation），经过 20 年的研究，数据协调获得了很大的发展。2003 年，Nguyen 等[3]首次将 Turbo 码作为基础码完成基于 Cascade 协议的 CVQKD 的协调，但协调效率不高。2005 年~2008 年，Bloch 等[1,4]在反向协调的方式下联合 MLC/MSD 多电平编译码，实现 25 km 的连续变量量子密钥分发系统，采用 LDPC 码为分量纠错码，协调效率达到 88.7%。上海交通大学曾贵华等在 GPU 上实现了码长为 10 000 的多维数据协调，协调速率虽然达到了 25 Mbit/s，但是码长过短，使协调效率不是很高[5]。文献[6]在 GPU 和 CPU 平台上分别实验对比等量数据的 LDPC 的译码算法的译码吞吐速率，结果表明 GPU 上的吞吐量是 CPU 的 9 倍。

目前，CV-QKD 系统的主要瓶颈在数据协调方向主要有以下 2 点。

1) 码长较短时，如目前传统通信中最长的码长为 10 000，由于高斯信道的最小纠错信噪比较高，导致协调效率不高，限制了 CV-QKD 系统的通信距离[7]。

2) 根据香农编码理论，码长越长，信道纠错编码的最小纠错信噪比越低。据研究报道和实验结果说明，码长为 200 000 可以使 CV-QKD 数据协调在低信噪比情况下实现收敛，而更长的码长对于减小最小纠错信噪比和提高协调效率增益发展潜力不大[4,8]。但如此长的码长在传统通信中极其罕见，导致协调计算量巨大，计算速度缓慢，通信时延太长，从而会限制连续变量量子密钥分发系统的实用性。

基于上述计算速度缓慢的问题，本文采用 GPU 对基于 LDPC 的 SEC 协议下数据协调算法进行并行化加速运算，并提出用静态双向十字链表存储数据结构 LDPC 码的稀疏校验矩阵；用 CUDA C 语言实现数据协调算法的并行译码，提高译码速率。

2 基于 LDPC 码的连续变量量子密钥分发的数据协调

2.1 连续变量量子密钥分发的数据协调

基于分层错误纠正协议(SEC)[9]的 CV-QKD 数据协调的主要过程：Alice 通过量子信道[10]将 X 发送给 Bob 的数据串 Y 是服从高斯分布的连续变量，X 与 Y 之间的互信息量为

$$I(X;Y)=H(X)+H(Y)-H(X,Y) \tag{1}$$

Bob 端通过最优量化[8]将 Y 变成离散变量 $\hat{Y}$，量化后的 X 与 $\hat{Y}$ 互信息量为

$$I(X;\hat{Y})=H(X)+H(\hat{Y})-H(X,\hat{Y}) \tag{2}$$

最优量化后的互信息量需满足 $I(X;\hat{Y})\leqslant I(X;Y)$。对实数采用等间隔的 4 bit、16 V 电平的编码压缩分为四级二进制码($L_1:L_4$)，各级信息会受到不同程度的噪声干扰，由 MSD 理论可知，L_1、L_2 在信噪比低的情况下互信息几乎为 0，因此这两级可以完全公开，不参与译码。L_3、L_4 级根据最优的码率进行译码，采用不同码率的校验矩阵，计算出两级序列的校验子 S_3、S_4，实现 SW 压缩编码[8]。L_1、L_2、S_3、S_4 作为协调信息由 Bob 经过公开理想信道送给 Alice。Alice 收到协调信息后结合边信息进行译码，完成整个过程的协调。

2.2 基于 LDPC 的 MSD 译码算法

MSD 译码方法就是各级分别译码，每一级的译码结果对其他级的译码有指引作用。级间迭代就是在变量节点到校验节点之间传递的外信息的基础上叠加节点内部流动的内信息。连续变量 Y_i 量化后形成的变量节点集表示为 $O_{(i)}$，边信息表示为 $O_{(ij)}$，设当前变量节点为 ij，与之相连的校验节点集为 $N(ij)$，其中，一个校验节点 K，K 相连的变量节点集为 $M(k)$，$i'jk$ 表示除变量节点 ij 外与校验节点 K 相连的所有变量节点的集合，$k'ij$ 表示除校验节点 K 外与变量节点 ij 相连的所有校验节点的集合。消息传递过程为变量节点传向给校验节点的消息 v_{ijk}，校验节点传向变量节点的消息 u_{kij}。S_k 表示校验序列中对应的校正子。所有的变量节点经过多次迭代的译码结果提供了内信息 e_{ij} 传递给其他级，供其他级使用，t 为迭代次数。图 1 用 Tanner 图表示和积置信传播算法中级间迭代的消息传递，黑色为外信息传递，灰色为内信息传递。

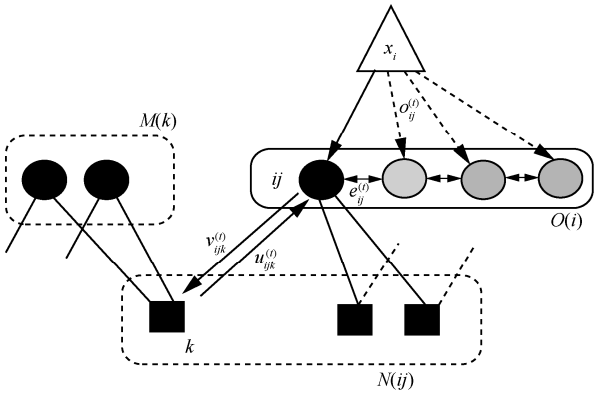

图 1 级间迭代的消息传递 Tanner 图

LDPC 码译码算法流程如下。

对于$1\leqslant t\leqslant t_{\max}$级间迭代的次数，$P$ 为 LDPC 译码迭代次数。

1) 初始化

初始化所有信息为零，即 $\forall i\in\{1,\cdots,l\}$, $\forall j\in\{1,\cdots,m\}$, $m=4,\forall k\in N(ij)$，$v_{ijk}^{(0)}=u_{kij}^{(0)}=0$

$$O_{ij}^{(0)}=\ln\frac{\sum\limits_{a}P(\hat{y}_{ij}=1\left|y_{i1},y_{i2},x_i\right.)}{\sum\limits_{a'}P(\hat{y}_{ij}=0\left|y_{i1},y_{i2},x_i\right.)}=\ln\frac{\sum\limits_{a}\int_{\tau_{a-1}}^{\tau_a}\frac{1}{\sqrt{2\pi\sigma}}\mathrm{e}^{-\frac{(y-x_i)^2}{2\sigma^2}}\mathrm{d}y}{\sum\limits_{a'}\int_{\tau_{a'-1}}^{\tau_{a'}}\frac{1}{\sqrt{2\pi\sigma}}\mathrm{e}^{-\frac{(y-x_i)^2}{2\sigma^2}}\mathrm{d}y} \tag{3}$$

其中，$a=\hat{y}_j=1$, $a'=\hat{y}_j=0$，$[\tau_a,\tau_{a-1}]$是令 $\hat{y}_j=1$ 时量化区间的上下限值，$[\tau_{a'},\tau_{a'-1}]$是令 $\hat{y}_j=0$ 时量化区间的上下限值。

2) 校验节点消息处理

校验节点 K 传递给变量节点 ij 的外消息

$$u_{kij}^{(t,p)}=2\tanh^{-1}\left[(1-2S_k)\prod_{i'\in\mathcal{M}(k)\backslash i}\tanh\left(\frac{v_{i'jk}^{(t,p-1)}}{2}\right)\right] \tag{4}$$

3) 变量节点消息处理

变量节点 ij 传递给校验节点 K 的外消息

$$v_{ijk}^{(t,p)}=\sum_{k'\in\mathcal{N}(k)\backslash k}u_{k'ij}^{(t,p)} \tag{5}$$

4) 译码判决

对所有变量节点计算硬判决消息

$$\lambda_{ij}^{(t)}=\sum_{k'\in\mathcal{N}(ij)}u_{k'ij}^{(t,p_{\max})}+O_{ij}^{(0)} \tag{6}$$

$$\hat{y}_{ij}=\begin{cases}0,\lambda_{ij}^{t}+O_{ij}^{0}\geqslant 0\\ 1,\lambda_{ij}^{t}+O_{ij}^{0}<0\end{cases} \tag{7}$$

5) 级间迭代

将 $\hat{y}_{ij}$ 代入式(6)和式(7)，令 $j'=4$，引入同一时隙所有层的硬信息，可得

$$O_{ij'}^{(t)}=\ln\frac{\sum\limits_{a}P(y_{ij'}=1\left|\hat{y}_{ij},y_{i2},y_{i1},x_i\right.)}{\sum\limits_{a'}P(y_{ij'}=0\left|\hat{y}_{ij},y_{i2},y_{i1},x_i\right.)}=\ln\frac{\sum\limits_{a}\left[erf\left(\frac{\tau_a-x_i}{\sqrt{2}\sigma}\right)-erf\left(\frac{\tau_{a-1}-x_i}{\sqrt{2}\sigma}\right)\right]}{\sum\limits_{a'}\left[erf\left(\frac{\tau_{a'}-x_i}{\sqrt{2}\sigma}\right)-erf\left(\frac{\tau_{a'-1}-x_i}{\sqrt{2}\sigma}\right)\right]} \tag{8}$$

通过不断地级间迭代，满足收敛或最大迭代次数译码结束。

3　基于 GPU 的数据协调并行化计算

3.1　GPU 概述

GPU 具有并行处理结构，可以实现多线程数据计算。被看作成一个高性能的计算设备，适合基于桌面平台进行大规模超级计算，完成海量数据的实时分析与处理，更强大的 GPU 甚至可以达到集群服务器的处理能力。这样，GPU 通用计算(GPGPU)可以加速处理如科学计算、大型数据库操作、实时信号处理等非图像问题，显示出巨大的潜力。

CUDA 用来协调 CPU 和 GPU 这 2 种处理单元的并发计算。与经典的 GPGPU 一样，CUDA 架构面向的是一个由 CPU 和 GPU 组成的异构计算网络，是由 CPU 控制和协调的[11]。CUDA 的程序按执行地点的不同分为主机函数和内核函数。主机函数是在 CPU 上调用和执行的，内核函数是在 GPU 上执行的，但可以从 CPU 或者 GPU 上调用。主机函数控制着整个程序的执行和 GPU 的启动、暂停。内核函数只是在 GPU 启动之后，在 GPU 上加速运算的部分，运算完后的数据结果需要传递回主机函数。基于 CUDA 的 GPU 计算是由 NVIDIA 公司提出的一种计算机实现平台，这是一种基于 C 的高级语言：CUDA C。CUDA C 是含有 NVIDIA 扩展和限制的类 C 语言，支持大多数 C 语言的指令和语法[11]。

3.2　数据协调并行化计算

基于 MSD 协调过程的重点是相关信源的 Slepian-Wolf(SW)译码[8,12]。LDPC 码的和积置信传播算法需要多次迭代才能收敛，每次迭代计算中大量的行运算和列运算，设 LDPC 码 C 码长为 N，其校验矩阵 $\boldsymbol{H}$ 的维数是 $M\times N$，每一行对应一个校验方程，即按式(4)进行行运算，共有 M 个校验方程；每一列对应码字的一位，共有 N 个码字，按式(5)进行列运算。在译码的一次迭代过程中，消息需要在变量节点之间和校验节点之间互相传递，且这种传递彼此之间互不影响。M 和 N 分别是 10^4 和 10^5 量级，因此有大量的重复计算，因此符合 SPMD(single program multiple data)并行计算模式，可以将此部分分配到 GPU 上执行。而 Bob 端的接收、量化、分级等单次运算可以分配到 CPU 上执行。图 2 表示 CV-QKD 并行数据协调方案。

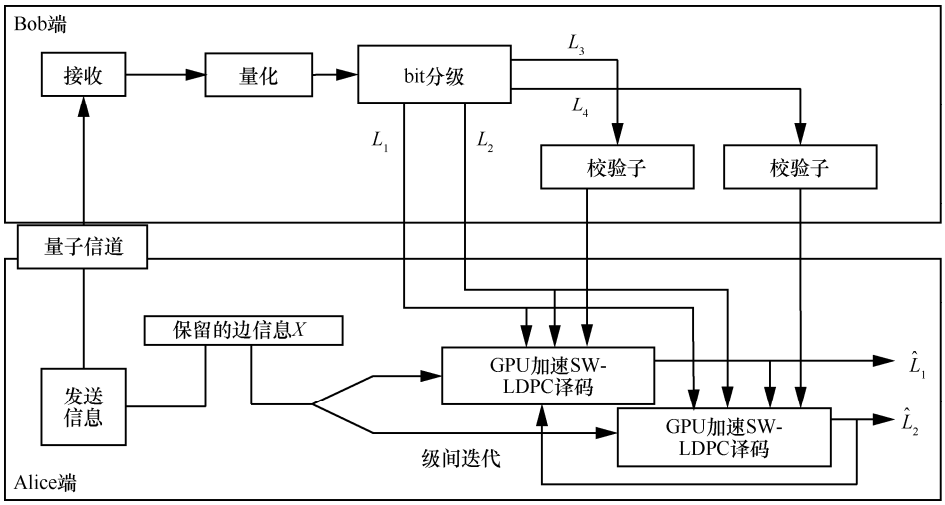

图 2 CV-QKD 并行数据协调方案

图 3 是 CPU 与 GPU 中的逻辑架构进行了对比[13]。其中，control 是控制器、ALU 是算术逻辑单元、cache 是 CPU 内部缓存、DRAM 就是内存。可以看到 GPU 设计者将更多的晶体管用于执行单元 ALU，而不是像 CPU 那样用于复杂的控制单元和缓存，这样就允许将大量的行运算线程和列运算线程分配在这些执行单元中，从而加速数据协调运算。

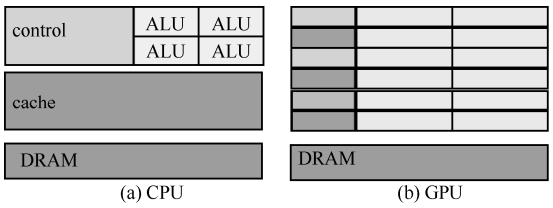

图 3 CPU 和 GPU 硬件逻辑结构对比

主机程序在 CPU 上执行，主要工作是将 LDPC 稀疏校验矩阵以静态双向循环十字链表的方式进行存储，并通过 CPU 与 GPU 之间的通信接口将该 LDPC 稀疏校验矩阵发送至 GPU；控制发送端 Alice 将大规模连续变量量子 X 发送至接收端 Bob，接收端 Bob 探测到 Y 序列后，对序列 Y 进行量化、分级，并得到校验子 $S_j(j=3,4)$，再将校验子发送回 Alice 发送端以进行 SW 译码。

本文的并行化采用英伟达公司提出的 CPU+GPU 混合编程框架 CUDA (compute unified device architecture)协议，基于 CUDA 协议的并行化 SW 译码算法如下。

1) 主机程序利用 cudaMalloc 分配函数为输入输出的数据对象分配 GPU 设备内存，输入数据对象主要包括校验矩阵的信息、变量节点和校验节点的似然比信息值、校验子信息、级间信息值。输出数据对象主要为变量节点的似然比信息。校验矩阵以 struct sparseMatrix ***H*** 的形式作为内核函数的输入参数，即校验矩阵信息可以像动态链表那样进行遍历访问。

2) 用 CUDA 定义的接口函数 cudaMemcpy 将已经在主机端赋值成功的输入数据复制到 GPU 分配的内存中。复制时 cudaMemcpy 的方向参数设置为主机端到设备，即用 cudaMemcpyHost ToDevice 复制数据。

3) 主机程序启动内核程序。GPU 上启动的是 3 个内核函数：初始化、校验节点消息处理、变量节点消息处理。

初始化信息。首先，准备数据对象，将校验矩阵按列划分为 N 个数据单元。其次，启动内核函数，实现线程数目的合理分配。CUDA 提供的启动函数语法是：内核函数名<<<线程块数,线程数>>>(输入输出参数)。GPU 中并行线程是按块分配的，每一个线程块均包含多个线程，执行相同的指令，达到线程块内的并行。不同型号的 GPU 中线程块中包含的线程数不同，如计算机的显卡上 GPU 每线程块拥有的线程数是 512，则需要并行执行的总线程数为 N，需要分配的线程块数为 $\frac{N}{512}$。同理，将校验矩阵按行划分为 M 个数据单元，并分配 $\frac{M}{512}$ 线程块。

执行内核的每个线程都会被分配一个独特的线程 ID，在内核函数中，通过 threadIdx 变量在内核中索引和访问线程 ID，每一列或行的信息被赋值给一个 threadIdx 值，实现每个线程串行地处理矩阵中一

列或行上的非零元素，按初始化计算式更新信息。N（或 M）个线程则并行处理校验矩阵所有列或行上的非零元素，并将计算结果也写入到输出对象的内存中。

校验节点消息处理。准备好各行数据对象并复制，将矩阵行信息赋给 threadIdx 值，对 $\boldsymbol{H}$ 矩阵行中指示的非零元素对应的消息进行访问，并按式(4)更新消息。

变量节点消息处理。处理过程与校验节点雷同，准备好各列数据对象并复制。将矩阵列信息赋给 threadIdx 值，对 $\boldsymbol{H}$ 矩阵的列中指示的非零元素对应的消息进行访问，并按式(5)更新消息。不同的是需要再加入一个级间信息的输入参数，并按式(8)完成级间迭代。

内核函数的运算主要是将输入参数中的值更新并且自动地存储到 GPU 的内存中，下一个内核函数输入参数获取的值就是已经更新好的值。每一个内核程序的执行由 CPU 控制串行的执行。

4) 将已经更新完成的变量节点的似然比信息值再通过 cudaMemcpy 函数复制到 CPU 的内存中，复制方向参数将重新设置为设备到主机端 cudaMemcpyDeviceTo Host。而后主机程序根据传递回来的似然比信息值进行译码判决，满足迭代条件则退出迭代过程。

4 大规模 LDPC 校验矩阵的静态双向十字链表存储方式

4.1 静态单链表的存储结构

信息论的研究结果表明，稀疏校验矩阵 $\boldsymbol{H}$ 越大，1 的分布就越随机，LDPC 的解码性能越好，越接近于香农极限。因此，本文采用的码长为 2×10^5 的稀疏校验矩阵。在 GPU 上实现译码算法，最核心的技术问题是 $\boldsymbol{H}$ 矩阵的存储。稀疏校验矩阵 $\boldsymbol{H}$ 阶数很大，1 的位置或个数经常变动，链表的存储结构可以较好地解决此类问题。但本文选用的支持并行运算的 CUDA 语言并不支持指针数据类型的算法语言，在 GPU 与 CPU 之间数据的传递只能分配固定大小的数据，因此动态链表就不适合在 GPU 上实现。针对这些问题，本文提出了一种只记录 1 的位置方式的静态双向循环十字链表的方法，充分结合了静态顺序存储和链式存储结构两者的优点，以静态顺序存储结构存储数据，但是每一个非零元素之间的逻辑关系是使用节点的数组下标作为位置索引来维持节点与节点之间前驱和后继的关系，这种数组下标就类似于动态链表中的指针，用来指向下一个节点的位置。下面以一个简单的例子说明一条静态单链表的存储结构。

例如，若存储的数据结构为

“1”→“2”→“3”→“4”→“5”→“6”→“7”→“8”

则存储结构如表 1 所示。

表 1　　静态链表的存储结构

地址	排序	索引
A[0]	2	6
A[1]	4	2
A[2]	5	7
A[3]	8	NULL
A[4]	1	0
A[5]	7	3
A[6]	3	1
A[7]	6	5

索引存储的位置信息，头节点 1 存储在 *A*[4]的位置，头节点中包含下一个节点 2 的位置，是 *A*[0]，那么索引就为 0；同理，节点 3 存储在 A[6]，在节点 2 中索引就为 6。

4.2 静态双向十字链表数据结构

CPU 上首先获取到 LDPC 稀疏校验矩阵中的非零元素的个数以及行列的信息，申请静态大小为非零元素个数的连续内存，并将所有的非零元素存储在这片内存中。然后以结构体数组的形式定义数据域，其中，数据域代表非零元素的节点，数据域里面的成员有似然比信息值及前后左右节点的位置地址信息值，位置地址信息值：*up* 表示该列的上一个节点的行信息；*down* 表示该列的下一个节点的行信息；*left* 表示该行的上一个节点的列信息；*right* 表示该行的下一个节点的列信息。位置地址信息值区别于动态链表中的指针类型，而是被定义为 int 的数据类型；第 i 个非零元素存放在静态内存中，在行上，i 中的右位置地址信息值代表第 i+1 个非零元素的行位置，左位置地址信息值代表第 i−1 个非零元素的行位置；在列上，i 中的前位置地址信息值代表第 i+1 个非零元素的列位置，后位置地址信息值代表第 i−1 个非零元素的列位置；静态双向十字链表的数据结构可定义如下

```
Typedef  struct  entry
{
        int   row, col;
        int   left, right, up, down;//存储 4 个方
```

向下一个节点的信息；

```
        double   pr, lr; //似然比信息值
};
Typedef  struct  sparseMatrix
{
        int      n_rows,n_cols;
        int      nnz;
        entry    *rows; //行链域
        entry    *cols;  //列链域
        entry    *mem; //非零元的存储位置
};
```

图 4 中的 $\boldsymbol{H}$ 矩阵中只记录了非零元素 1 的位置 (i, j)，相邻的非零元素之间的 0 直接省去，*rows*、*cols* 表示 2 个链域：行链域和列链域，其指针指向申请的非零元素大小的静态内存的地址。行链域上，*i* 行上每一个非零元素 1 的列信息是不一样，*cols*[*i*].*left* 和 *cols*[*i*].*right* 存储的值分别指向前一个节点和后一个节点的位置信息。列链域上，*j* 行上每一个非零元素 1 的行信息是不一样，*rows*[*j*].*up* 和 *rows*[*j*].*down* 存储的值分别指向前一个节点和后一个节点的位置信息。这样构成了十字双向静态的链表，提高了非零元素的定位速度，也大大降低了存储内存。

5 仿真结果及分析

本文使用的是具有英伟达 Geforce GT 650 MB 的 GPU、2.5 GHz Intel Core i5-3210M 双核 CPU 和 8 GB 内存的普通笔记本电脑，选择 2×10^5 的码长，分为 10 的分组，迭代次数设为 100 进行统计。

根据信道容量规则[14,15]和最佳码率公式得出最佳码率值，信噪比 R_{SN}=3 的最佳码率为 0.002/0.016/0.3/0.95[16]，前两级不参与译码直接公开。本文采用 Mackay[17]等 2B 构造方式产生校验矩阵，码长为 2×10^5，对比 CPU 和 GPU 运算的实验结果如表 2 所示。

从表 2 可以看出，本文方案与文献[5]方案采用的方法在译码速率上有差别，主要原因是采用的码长之间相差 20 倍，而文献[5]方案采用多维数据协调的方式进行协调，不需多级译码，译码次数会减少很多。显卡的计算能力不同，文献[5]采用的 GPU 型号是 GeForce GTX Titan Black，计算能力比大约是 8:5。根据 Mackey 关于 LDPC 码译码速度与码长的关系为 $6Nt$，其中 N 为码长，t 为迭代次数，$\dfrac{25\,000}{16.4}=8\times\left[\left(6\times\dfrac{20\,000}{100}\right)\times\dfrac{8}{5}\right]$，即本文方案的速率是文献[5]方案的 8 倍。考虑到迭代次数因素和协调方式的不同，指标的差别是在合理范围之内的。

在数据协调过程中会受到香农极限的限制，导致协调效率不可能达到 100%。协调效率 β 的估算式为

$$\beta=\frac{H(\hat{Y})-m+\sum_{i=1}^{m}R_s^i}{I(X;Y)} \tag{9}$$

其中，$H(\hat{Y})$ 为 Bob 接收到 Y 量化为 $\hat{Y}$ 的信息熵，$I(X;Y)$ 为 Alice 与 Bob 之间的互信息，m 为级数，

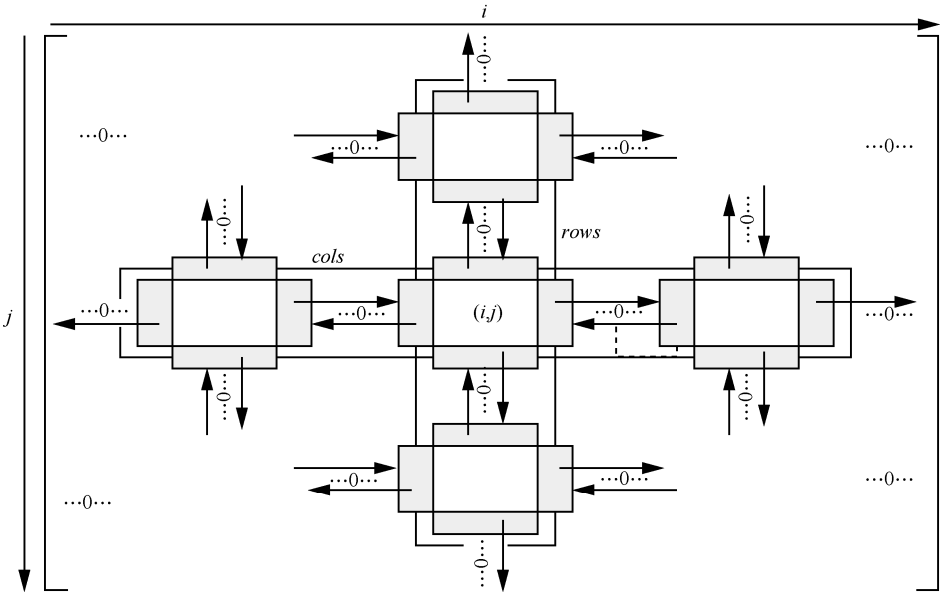

图 4 $\boldsymbol{H}$ 矩阵中非零元素十字双向链表结构

表 2　　在 CPU 和 GPU 执行译码速度的对比

方案	码长	每一级码率	收敛信噪比 R_{SN}/dB	100 次迭代的时间/s	译码速率/(kbit·s^{-1})	协调效率
GPU 并行	200 000	0/0/0.3/0.95	4.9	12.28	16.4	0.917 1
仅 CPU	200 000	0/0/0.3/0.95	4.9	185.76	1.07	0.917 1
上海交大[5]	10 000	未报道	未报道	未报道	25 000	—

R_s^i 为各级的码率。由表 2 和式(9)可知，在译码收敛后，在 $I(X;Y)$ 一定的情况下，量化越好，各级码率越高，则协调效率越高，但是协调效率的提高限制了译码性能。只有保证译码性能和协调效率的前提下，译码速度的提高才会有意义。本文采用 GPU 并行加速比 CPU 不并行译码速度提高了 15 倍。

6　结束语

在 GPU 平台上对 SEC+MLD 框架下的连续变量量子密钥分发逆向数据协调进行了仿真，采用静态双向十字链表的数据结构存储 LDPC 码的校验矩阵，在连续变量为 2×10^5、收敛信噪比为 4.9 dB、协调效率为 91.71%的前提下、译码速度为 16.4 kbit/s。

降低收敛信噪比、提高协调效率和译码速率，对提高密钥的传输距离有重要的影响。下一步将在低信噪比的情况下，通过 GPU 的优化实现高效率的协调。

参考文献：

[1] BLOCH M, THANGARAJ A, MCLAUGHLIN S W. Efficient reconciliation of correlated continuous random variables using LDPC codes[J]. Arxiv preprint cs, 2005: 0509041.

[2] BRASSARD G, SALVAIL L. Secret-key reconciliation by public discussion[C]//The Workshop on Advances in Cryptology-Eurocrypt. 1993:410-423.

[3] NGUYEN K C, ASSCHE G V, CERF N J. Side-information coding with turbo codes and its application to quantum key distribution[J]. Mathematics, 2004.

[4] LODEWYCK J, BLOCH M, GARCIA-PATRON R, et al. Quantum key distribution over 25 km with an all-fiber continuous-variable system[J]. Physical Review A, 2007, 76(4): 538-538.

[5] LIN D K, HUANG D, HUANG P, et al. High performance reconciliation for continuous-variable quantum key distribution with LDPC code[J]. International Journal of Quantum Information, 2015, 13(2): 1550010.

[6] GOLUB G H, VAN L,CHARLES F. Matrix Computations[M].北京:科学出版社,2009.
GOLUB G H, VAN L,CHARLES F. Matrix Computations[M]. Beijing: Science Bress, 2009.

[7] 郭大波, 刘纲, 张宁. 量子高斯密钥分发的逆向数据协调[J]. 量子光学学报, 2013, 19(3): 219-226.
GUO D B, LIU G, ZHANG N, et al. Reverse Reconciliation of Quantum Gaussian Distributed Key [J]. Acta Sinica Quantum Optica, 2013, 19(3): 219-226.

[8] 郭大波, 张彦煌, 王云艳. 高斯量子密钥分发数据协调的性能优化[J]. 光学学报, 2014, 34 (1): 225-231.
GUO D B, ZHANG Y H,WANG Y Y. Performance optimization for the reconciliation of Gaussian quantum key distribution[J]. Acta Optica Sinica, 2014, 34 (1): 225-231.

[9] VAN ASSCHE G, CARDINAL J,CERF N J. Reconciliation of a quantum-distributed Gaussian key[J]. Information Theory IEEE Transactions on, 2002, 50 (2): 394-400.

[10] 白增亮, 王旭阳, 杜鹏燕, 等. 连续变量量子密钥分发的数据逆向协调[J]. 量子光学学报, 2012, 18 (1): 23-26.
BAI Z L, WANG X Y, DU P Y, et al. Reverse reconciliation for continuous variable quantum key distribution[J]. Acta Sinica Quantum Optica, 2012, 18 (1): 23-26.

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

[12] 郭大波, 张宁,刘纲. 基于 Turbo 码的量子高斯密钥分发的数据协调[J]. 量子光学学报, 2013, 19 (1): 32-38.
GUO D B, ZHANG N, LIU G. Reconciliation of quantum Gaussian distributed key based on Turbo codes[J]. Acta Sinica Quantum Optica, 2013, 19 (1): 32-38.

[13] DAVID B K,WEN M, HWU W.大规模并行处理器编程实战[M].北京:清华大学出版社,2013.
DAVID B K,WEN M, HWU W. Programming massively parallel processor[M]. Beijing: Tsinghua University Press, 2013.

[14] JABER BORRAN M,AAZHANG B. Multilevel codes and iterative multistage decoding: rate design rules and practical considerations[C]// Wireless Communications and Networking Conference, 2000: 36-41.

[15] WACHSMANN U, FISCHER R F H,HUBER J B. Multilevel codes: theoretical concepts and practical design rules[J]. IEEE Transactions on Information Theory, 1999, 45 (5): 1361-1391.

[16] BLOCH M, THANGARAJ A, MCLAUGHLIN S W, et al. LDPC-based secret key agreement over the Gaussian wiretap channel[C]//IEEE International Symposium on Information Theory.2006: 1179-1183.

[17] MACKAY D J C, NEAL R M. Near Shannon limit performance of low density parity check codes[J]. Electronics Letters, 1997, 33 (6): 457-458.

作者简介：

刘绍婷（1991-），女，山西吕梁人，山西大学硕士生，主要研究方向为量子密钥分发。

王晓凯（1963-），男，山西运城人，博士，山西大学教授，主要研究方向为通信网络管理、控制与优化等。

郭大波（1963-），男，山西阳泉人，博士，山西大学副教授，主要研究方向为量子密钥分发。