基于级联编码的区块链分片存储方案

田有亮[1,2,3,4]，黄钰清[1,2,3,4]，王帅[1,2,3,4]

（1.贵州大学公共大数据国家重点实验室，贵州 贵阳 550025；2.贵州大学计算机科学与技术学院，贵州 贵阳 550025；
3.贵州大学密码学与数据安全研究所，贵州 贵阳 550025；4.贵州省密码学与区块链技术特色重点实验室，贵州 贵阳 550025）

摘　要：传统区块链存在存储可扩展性难题，现有研究基于纠删码理论降低了区块链存储开销，但在区块的译码恢复时会带来高额的计算与通信开销。为解决以上难题，提出了一种基于级联编码的区块链分片存储方案。通过添加预编码层改进现有无码率纠删码，实现了$O(n)$的编译码复杂度。考虑译码过程中节点间的通信时延差异，提出了基于Metis的时延感知分片算法，通过时延权重决定节点的分片归属，降低译码过程中的通信开销。仿真结果表明，所提方案不仅保证了区块链数据可靠性而且所需的计算开销与通信开销也优于传统方案。

关键词：区块链；存储可扩展性；纠删码；分片技术

中图分类号：TP302

文献标志码：A

DOI：10.11959/j.issn.1000–436x.2024114

Blockchain sharding storage scheme based on concatenated coding

TIAN Youliang[1,2,3,4], HUANG Yuqing[1,2,3,4], WANG Shuai[1,2,3,4]

1. State Key Laboratory of Public Big Data, Guizhou University, Guiyang 550025, China
2. College of Computer Science and Technology, Guizhou University, Guiyang 550025, China
3. Institute of Cryptography & Data Security, Guizhou University, Guiyang 550025, China
4. Guizhou Provincial Key Laboratory of Cryptography and Blockchain Technology, Guiyang 550025, China

Abstract: Traditional blockchain faces the challenge of storage scalability. Existing research has reduced the storage overhead of blockchain based on erasure coding theory, but it brings high computational and communication spending during the decoding and recovery of blocks. To solve these problems, a blockchain sharding storage scheme based on concatenated coding was proposed. By adding a pre-coding layer to improve the existing rateless erasure code, a decoding complexity of $O(n)$ was achieved. Considering the communication delay skew between nodes during the decoding process, a delay-sensitive sharding algorithm based on Metis was proposed, which cut down the communication expenditure in the decoding process by delaying weights to determine the shard ownership of nodes. Simulation results show that the proposed scheme not only ensures the reliability of blockchain data, but also has lower computational and communication cost compared to traditional schemes.

Keywords: blockchain, storage scalability, erasure coding, sharding technology

---

收稿日期：2024–02–26；**修回日期：**2024–06–03

通信作者：田有亮，youliangtian@163.com

基金项目：国家重点研发计划基金资助项目（No.2021YFB3101100）；国家自然科学基金资助项目（No.62272123）；贵州省高层次创新型人才基金资助项目（No.[2020]6008）；贵州省科技计划基金资助项目（No.[2020]5017, No.[2022]065）；贵阳市科技计划基金资助项目（No.[2022]2-4）

Foundation Items： The National Key Research and Development Program of China (No.2021YFB3101100), The National Natural Science Foundation of China (No.62272123), The Project of HighLevel Innovative Talents of Guizhou Province (No.[2020]6008), The Science and Technology Program of Guizhou Province (No.[2020]5017, No.[2022]065), The Science and Technology Program of Guiyang (No.[2022]2-4)

0 引言

区块链是一种分布式的数字账本技术，自问世以来得到了社会各界的广泛关注[1]。为了实现去中心化，区块链采用全副本存储机制[2]，以确保在部分节点失效的情况下，其余节点仍能维护账本的完整性并保证系统正常运转。但高度冗余的存储机制会占用每个节点大量的存储空间，随着区块链交易数据量的快速增长，最终系统存储能力将达到瓶颈[3]。根据加密货币交易所统计的数据，截至2023年6月，完整的比特币区块链大小已经达到468 GB，并以每年50 GB的大小不断增长。可以预见，区块链的存储可扩展性将成为亟待解决的重要问题。

为解决这一问题，研究者最初提出了调整区块大小、引入轻量级节点等方案来缓解区块链节点的存储压力[4-5]，但是这些方案会导致区块链数据可靠性下降、隐私泄露等问题[6]。与之相同，区块链下的存储方案也能减轻节点的存储压力[7-9]，但同样引入了中心化风险和数据安全隐患。

近年来，有研究者通过将纠删码技术应用到区块链中来优化区块链节点的存储开销。相较于区块修剪、添加轻量级节点等传统方案[10-11]，纠删码技术可以将区块链账本分割成多个较小片段并存储在不同的节点中，节点的存储压力由完整的区块链账本变为部分账本片段，极大地降低了节点的存储开销[12]。同时，纠删码技术可以生成额外的校验片段，保证在某些节点出现故障或者被攻击的情况下，仍然可以通过其余节点提供的片段恢复原始数据，提高了数据的可靠性与完整性，增强了区块链系统的容错性。

然而，纠删码技术在解决区块链存储可扩展性难题的同时也带来了新的挑战。首先，纠删码技术是通过计算开销换取存储开销，在减少节点存储消耗的同时也带来了额外的计算开销[13]；其次，应用纠删码技术的节点在区块恢复过程中需要向其余节点请求数据片段，而分散的节点地理位置会带来高额的通信开销；最后，在节点数量动态变化的区块链网络中，由于传统纠删码技术通常采用固定的编解码策略，参与节点变更的同时需要频繁调整或更新编解码策略，因此，不利于区块链的存储可扩展性。

为解决上述挑战，本文提出了基于级联编码的区块链分片存储方案，该方案从优化区块链节点编解码的计算开销和系统可扩展性的角度出发，通过添加预编码层改进现有无码率纠删码，降低了节点编译码的计算开销，提高了系统可扩展性。同时，为了降低译码过程中的通信开销，提出了基于Metis的时延感知分片算法，以节点之间的通信时延作为综合权重对节点进行分片，降低节点的译码时间并优化整体区块链网络的通信开销。本文主要贡献如下。

1) 提出了基于级联编码的编译码方案，通过添加预编码层改进现有无码率纠删码，将方案的译码复杂度降至$O(n)$，提高了系统的可扩展性。

2) 提出了基于Metis的时延感知分片算法，以节点之间的通信时延作为综合权重进行分片，降低节点译码时请求编码块带来的通信时延并优化分片内节点的整体通信开销。

3) 实验结果表明，所提方案相较于传统方案具有更优的计算开销与通信开销。

1 相关工作

Perard等[12]提出了通过纠删码理论来存储区块链数据的方案，并引入了一种基于纠删码的新型存储节点，但该方案需要为每个生成的区块进行编码与分发，这会导致节点产生大量的计算开销。Wu等[14]提出了利用低密度奇偶校验（LDPC, low density parity check）码来减少区块链节点存储需求的编码方案，该方案对区块链中的多个区块数据进行编码，但是在恢复区块链时译码开销过大。Raman等[15]提出了一种基于纠删码的分布式编码方案，该方案结合秘密共享和分布式存储，使每个节点只存储交易的一部分，并使用动态区域分配增强数据的完整性，但同时也给节点带来了额外的存储开销。Qi等[16]提出将里所（RS, Reed Solomon）码应用于拜占庭容错（BFT, Byzantine fault tolerant）的联盟链中，实现了纠删码在拜占庭环境中的容错存储，但该方案在节点变更时需要进行重新编码，从而耗费大量的通信与计算开销，不利于区块链的存储可扩展性。Kadhe等[17]提出了一种基于安全喷泉码（FC, fountain code）的区块链存储体系结构（SeF, secure fountain），通过无码率LT（Luby transtorm）码的特性抵御恶意节点，实现了拜占庭容错、存储成本与区块译码成本之间的权衡，但该方案的译码成本随区块数量的增加呈对数增长，同时在区块链较大时不能实现对区

块链的良好覆盖而造成额外的译码开销。上述方案均存在一定缺陷，本文针对现有区块链编码方案中存在的计算开销问题，提出了一种基于级联编码的编译码方案，同时为实现节点交互过程中通信时延的负载均衡，提出了一种基于Metis的时延感知分片算法。

2　基础知识

2.1　无码率纠删码——LT码

喷泉码也被称为无码率码，即可以将K个原始数据包通过喷泉码编译码产生无限数量的编码数据包，接收端只需要收到其中任意的N（N略大于K）个编码数据包，就能以高概率恢复所有的原始数据包。喷泉码取名于其编码过程，类似一个喷泉源源不断地喷出水滴（编码数据包），用户只需要用杯子接收到足够的水就能饮用（译码成功），而不需要考虑杯子里的水由哪些水滴组成。喷泉码示意如图1所示。

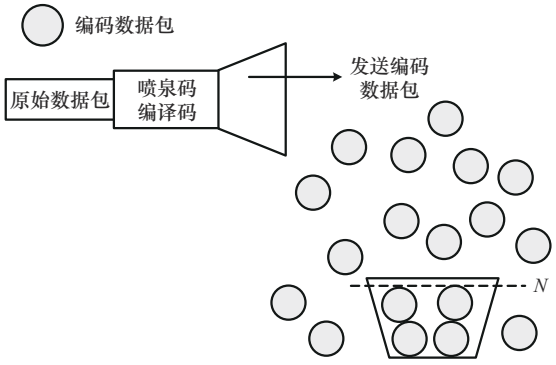

图1　喷泉码示意

LT码是第一种具有实用价值的喷泉码，由Luby[18]在2002年首次提出。它的编码和译码过程复杂度较低，能够以更少的编码块来恢复原始数据，这些特征使其适用于区块编码方案。LT编码过程如图2所示。首先LT码会将原始数据包划分为k个等长的原始数据块，之后利用度分布函数Ω_d随机均匀地选取d（$d \leqslant k$）个不同的数据块[19]，最后通过异或运算的方式得到一个编码块，d则被称为这个编码块的度值。LT编码的核心是选取一个合适的度分布函数，根据该函数产生随机度值的编码块，不同的度分布函数会对LT编码的性能带来极大影响，良好的度分布函数能够尽可能地降低编码块的平均度值，减少所需的异或运算次数。d值的大小会导致编码速率的不同，两者呈正相关，因此想要获得较快的编码速率，就需要选取较小的d值。同时d值的波动会影响原始数据的覆盖率，波动较大的d值可以保证所有的原始数据包参与编码，提高译码最终的成功率。因此若想将LT码应用于区块链存储，需要权衡译码成功率和编译码速率2个因素对节点的影响并选取合适的度分布函数。

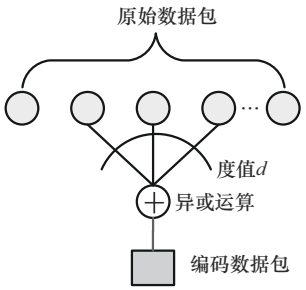

图2　LT编码过程

2.2　分片技术

为了克服传统的区块链架构在面对存储需求快速增长时的局限性，研究人员提出了区块链分片技术，旨在将整个区块链网络分割成多个较小的片段[20]。在区块链分片中，所有的交易和状态数据被分散存储在不同的分片中，而不是集中存储在整个网络中的每个节点上。这样有效地减少了每个节点需要存储的数据量，从而降低了存储成本和资源消耗。同时，分片技术还提高了整个网络的吞吐量和处理能力，因为每个分片可以并行地处理交易和计算[21]。最后，通过增加网络的容量和可扩展性，分片技术使更多的参与者能够加入区块链网络中而不会造成性能瓶颈。

目前，为解决引入分片技术带来的负载均衡和资源分配问题，研究人员提出了多种分片算法和共识机制。例如，基于共识机制的分片方案通过引入可验证随机函数（VRF, verifiable random function）等方法[22]来选择验证和打包交易的节点。而基于细粒度负载均衡的分片方案可以通过对区块链账本与账户进行分割来保证每个分片的负载均衡[23]。此外，还有一些基于可验证计算的分片方案[24]，可以将计算任务委托给其他节点来降低本地计算的压力。总体来说，区块链分片技术通过分割区块链账本，使每个分片内的节点只存储部分数据，提供了一个可行的方案来解决区块链的存储可扩展性问题。

3 方案模型

3.1 符号定义

在介绍方案模型之前，本文对一些符号和基本概念进行了定义，如表1所示。

表1 符号定义

符号	定义
e_i	第i个共识轮次
t_i	第i轮共识产生的区块
s_i	分割得到的第i个区块子链
hb	全局的区块头部链
Ω_d	度分布函数
d	通过度分布函数得到的度值
S	分片的数量
N_i	第i个分片的存储节点
B	原始块
IB	中间块
EB	编码块

3.2 系统模型

为解决传统区块链编码存储方案带来的弊端，本文方案通过添加预编码层对现有无码率纠删码进行改进以降低节点的计算开销，并设计分片算法优化节点的通信开销。图3展示了本文所提基于级联编码区块链分片存储模型。系统流程包含2个阶段与2个编码过程：分别为共识阶段和存储阶段，预编码过程和LT编码过程。其中参与实体包含客户端、共识委员会以及存储节点，客户端会发送常规交易请求和区块恢复请求两类请求。

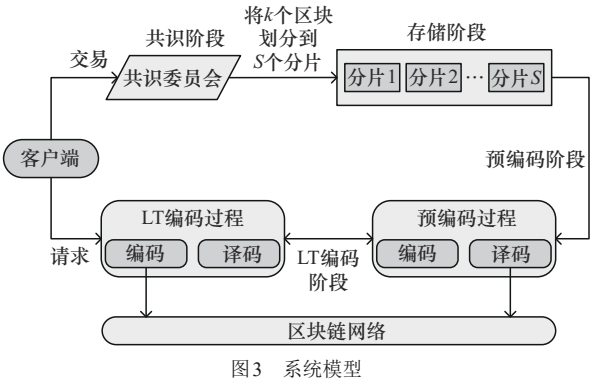

图3 系统模型

在共识阶段开始前，系统会通过随机方式选出一组节点构成共识委员会，并将除共识委员会之外的节点划为存储节点，存储节点通过分片算法划分为S个分片。系统以产生固定数量t个区块的时间作为一个共识轮次e，每个e对应一个共识阶段。在共识阶段过程中，共识委员会负责处理来自客户端的请求。对于常规交易请求，共识委员会在确认交易的合法性之后，将达成一致的交易添加进区块。在产生t个区块之后，共识委员会按照分片数量S将t分割为s个区块子链，每个区块子链包含k个原始区块B，即$s_i=\{B_1,B_2,\cdots,B_k\}$。之后共识委员会将不同的区块子链$s_i$广播到各分片$S_i$中，在收到$S_i$中$2f+1$个节点的响应时，共识委员会将认为$s_i$能正确交付到$S_i$中，当所有分片完成交付时，该轮共识结束，进入下一轮共识阶段。

在存储阶段，S_i中的存储节点N_i会对接收到的区块子链s_i进行编码并存储，根据分片内部的预编码策略与LT编码策略，N_i会将s_i中的k个原始块编码为$m(m>k)$个编码块EB，最后随机存储EB并删除接收到的区块子链s_i。需要注意的是，存储节点并不参与共识过程，但会在每个共识轮次结束后存储更新全局的区块头部链hb，hb保存了每轮共识中生成的所有区块哈希值、该区块所属子链、子链所划分到的分片等信息。节点可以通过hb快速定位客户端所请求的区块所在分片，并且在编码过程中节点不需要对编码块附加校验信息，通过比较译码过程中得到的区块哈希值即可鉴别恶意编码块并确保区块的正确性。对于来自客户端的区块恢复请求，节点会通过hb来判断该区块是否存储在自身所属分片。若是，则节点会在该分片内发起恢复请求，向其他节点请求编码块并进行译码恢复；若不是，则节点会将恢复请求发送到共识委员会，由共识委员会转发到相应的分片中。

为了实现拜占庭容错，在本文方案中节点会对每$N'=\dfrac{N-\left\lfloor\dfrac{N-1}{3}\right\rfloor}{1+\varepsilon}$个原始块进行编码并存储，其中，$N$为分片内节点数量，$\varepsilon$为级联编码的冗余开销。在发起区块恢复请求后，节点只需收到大于$N'(1+\varepsilon)$个诚实节点的编码块就会开始译码，当所接收的编码块不足以完成译码时，节点会向剩余节点请求更多编码块。

3.3 级联编码模型

LT码不需要确定码率，同时在译码过程中接收端不需要考虑丢包以及不用对接收到的数据包进行反馈这一特性，使其非常适用于区块链。但在基

于LT码的编码方案SeF[17]中，当需要编译的区块链较长时，度分布函数难以实现对所有区块的覆盖，导致部分区块不能参与编码从而增大译码失败的概率。因此该方案选择了译码复杂度低且在长码下性能优秀的LDPC码作为内码对原始区块进行预编码，通过校验矩阵生成额外的校验编码块，提高了原始区块的编码覆盖率。由于预编码的引入，作为外码的LT码可以选择更加简易的度分布函数来降低编译码复杂度。同时LDPC码良好的纠错能力能够确保节点收到消息的正确性，从而提高级联编码的译码成功率。级联编码过程如图4所示。

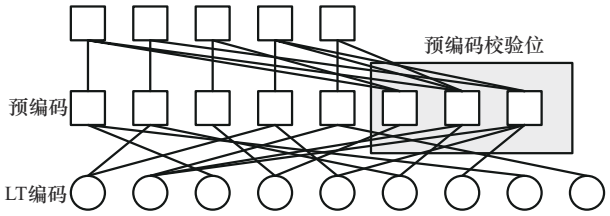

图4　级联编码过程

预编码过程与LT编码过程都包含编码与译码2个子模块，其中，编码过程代表区块的存储，节点N_i通过LDPC预编码将原始区块B编码为中间块IB，将IB作为LT编码的输入生成并存储部分编码块EB；译码过程代表区块的恢复，节点N_i收集分片内其他节点的EB，通过LT译码恢复中间块IB，最后通过LDPC译码将中间块IB恢复为原始块B。在区块的恢复过程中，方案允许中间块IB未被全部译出，只需译出一定比例的中间块IB就可通过预编码LDPC码的容错能力恢复出所有原始块B。

4　方案概述

4.1　基于Metis的时延感知分片算法

在区块链网络中，节点的地域分布与网络拓扑使得节点之间的通信时延存在差异。尤其在基于编码的区块链方案中，由于编码块分散存储于不同地域的节点中，节点发起区块恢复请求时带来了大量的通信开销。为了有效优化网络中的通信效率，降低区块恢复过程中的通信开销，本文提出了一种基于Metis的时延感知分片算法。该算法以节点之间的通信频率与通信时延作为综合权重，构建通信时延加权图，并通过Metis算法将节点划分为多个相互独立的分片。不同分片内的节点通过编码保存一部分区块链，在区块的恢复过程中，节点只需与同一分片内的其他节点进行通信即可恢复所需区块，不需要与其他分片进行交互。节点分片过程如图5所示。

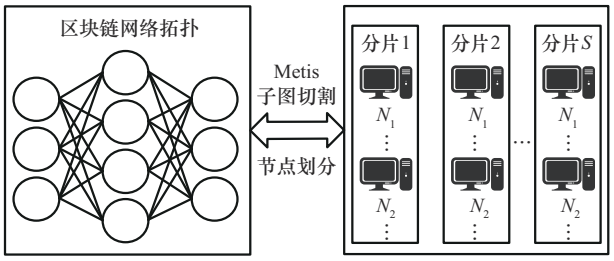

图5　节点分片过程

在分片开始前，根据节点集合$N=\{N_1,\cdots,N_n\}$及节点之间的通信关系构建区块链网络拓扑，通过网络拓扑测试节点之间的通信频率和通信时延，得到通信频率矩阵$\boldsymbol{M}_d$和通信时延矩阵$\boldsymbol{M}_\tau$，以d_{ij}和τ_{ij}表示节点i和j之间的通信频率和通信时延，通过综合权重函数$\alpha d_{ij}+(1-\alpha)\tau_{ij}$计算节点之间的综合权值$v_{ij}$，其中$\alpha$表示权重系数。随后根据节点及边权值构建加权图，并以$(n,e_1,v_1,\cdots,e_j,v_j)$的形式对节点的拓扑关系和边权值进行存储，其中$n$表示节点，$(e,v)$表示与该节点的邻接节点与边权值。最后，调用Metis算法对加权图进行划分，将N个节点划分到不同分片$S=\{S_1,\cdots,S_i\}$中，使每个分片节点之间的边权值之和最优，这意味着分片内的节点之间有着高通信频率和低通信时延。具体算法如算法1所示。

算法1　基于Metis的时延感知分片算法

输入 节点集合N，通信频率矩阵$\boldsymbol{M}_d$，通信时延矩阵$\boldsymbol{M}_\tau$，权重系数α，分片数量S

输出 分片结果

1) 初始化加权图G
2) 当节点$i=1,2,\cdots$时，开始循环1
3) 当节点$j=1,2,\cdots$时，开始循环2
4) 根据$\boldsymbol{M}_d$和$\boldsymbol{M}_\tau$得到d_{ij}和τ_{ij}，计算i，j之间的综合权值$v_{ij}=\alpha d_{ij}+(1-\alpha)\tau_{ij}$
5) 得到节点i的拓扑关系及边权值(e_{ij},v_{ij})
6) 更新加权图G的拓扑关系和边权值
7) 结束循环2
8) 结束循环1
9) 执行图划分算法，将N个节点划分到S个分片中
10) 返回分片结果

对于传统的区块链编码方案来说，由于节点广泛分布于不同地理位置，在区块译码恢复过程中，节点向其他节点请求编码块时会带来大量的通信开销。而本文将分片技术应用于编码方案，使同一分片内的节点共同编码存储一部分区块子链，分片内的节点只需要与其内部的其他节点进行交互即可完成区块的恢复请求，从而避免了跨分片的数据传输，均衡了各分片间节点的通信开销，更加适用于编码方案区块链的分布式存储场景。

4.2 区块生成及分割

在本文方案的共识过程中，每个共识轮次e结束时都会产生t个区块，用$\{b_1,b_2,\cdots,b_t\}$表示共识轮次e_i中生成的t个区块，如图6所示。

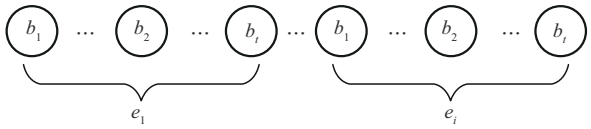

图6 区块生成示意

随后，对e_i中生成的连续t个区块按照分片数量S进行分割。为了简单起见，以连续的$k=\left\lfloor\frac{t}{s}\right\rfloor$个区块为单位对原始块进行分割，对于不满足长度$k$的子链以空白区块作为填充，最后得到对应每个分片的原始区块子链s_i，如图7所示。

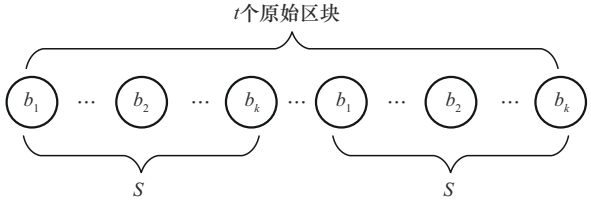

图7 区块子链分割示意

4.3 区块编码

方案的编码过程分为2个阶段，当分片S_i中的节点N_i收到区块子链s_i后，会对s_i进行LDPC预编码来生成中间块IB，并通过LT编码生成并存储部分编码块EB，如算法2所示。

算法2 区块级联编码

输入 分片S，节点集N，共识轮次e，区块子链s

输出 编码块EB

1) 对于分片S_i，当$i=1,2,\cdots$时，开始循环1
2) 对于节点集N_i，当$i=1,2,\cdots$时，开始循环2
3) 节点i通过校验矩阵对s进行编码操作生成校验块C并得到中间块$\mathrm{IB}=\{B^e|C^e\}$
4) 通过度分布函数生成度值d
5) 从IB中选取d个区块进行异或操作，得到编码块EB^e
6) 节点i编码结束，向共识层返回响应
7) 结束循环2
8) 如果共识层收到分片S_i的$2f+1$个响应，则分片S_i编码阶段完成
9) 分片S_i中的节点存储EB^e并删除s和IB
10) 结束循环1

在预编码阶段，节点会使用系统形式的LDPC码通过校验矩阵$\boldsymbol{H}$对区块子链$s_i=\{B_1,B_2,\cdots,B_k\}$进行快速编码，其优点在于编译后得到的中间块IB包含了原始块的信息，方便节点利用区块头部链中的哈希值来检验编码块是不是恶意生成的。IB由原始块B与校验块C组成，规定编码后校验块C位于原始块B之后，即$\mathrm{IB}=[B|C]$，其次，将校验矩阵$\boldsymbol{H}(k\times m)$写为$\boldsymbol{H}=[\boldsymbol{H}_1|\boldsymbol{H}_2]$，其中，$\boldsymbol{H}_1$为$k\times k$的矩阵，$\boldsymbol{H}_2$为$k\times(m-k)$的矩阵。根据$\mathrm{IB}\boldsymbol{H}^{\mathrm{T}}=0$，得到$\boldsymbol{H}_1X+\boldsymbol{H}_2C=0$，推导出$C=\boldsymbol{H}_2^{-1}\boldsymbol{H}_1B$，即求得校验位$C$。最后将校验位$C$与原始块$B$合并就得到了$m$个中间块IB，因为IB是通过系统形式的LDPC预编码生成，所以有$\{\mathrm{IB}_1,\cdots,\mathrm{IB}_m\}=\{|B_1,\cdots,B_k|,|C_{k+1},\cdots,C_m|\}$，即中间块的前$k$位对应原始块$B$，后$m-k$位对应校验块$C$。

由上述的预编码过程可知，所有的原始块参与了编码。此时节点将对IB进行二次编码。方案定义LT码参数为(IB,Ω_d)，其中，IB表示LT编码方案的输入，Ω_d表示系统预设的度分布函数。方案采用了Shokrollahi[25]提出的度分布函数，如式(1)所示。设$D=\left\lceil\frac{4(1+r)}{r}\right\rceil$，$\mu=\left(\frac{r}{2}\right)+\left(\frac{r}{2}\right)^2$，有

$$\Omega_d=\begin{cases}\dfrac{\mu}{\mu+1},d=1\\ \dfrac{1}{(\mu+1)d(d-1)},d=2,\cdots,D\\ \dfrac{1}{(\mu+1)D},d=D+1\end{cases}\tag{1}$$

编码过程如下。

1) 节点通过预先定义的度分布函数Ω_d生成

一个随机度值d。

2) 节点根据该度值d从中间块IB中随机选取d个中间块。

3) 将选取的d个中间块进行异或运算，从而得到一个编码块EB。

4) 重复上述操作，每次都会得到一个编码块EB，直到生成n个编码块。

5) 节点按照规则只保留部分编码块EB，并删除中间块与其余编码块并向共识层节点返回响应，当共识层收到一个分片内$2f+1$个响应时，则代表该分片的编码阶段完成。

4.4　区块译码

当分片S_i中的节点收到客户端对于原始区块的恢复请求时，需要向分片中的其他节点请求编码块并执行译码操作。在译码过程中，本文方案使用置信传播（BP, belief propagation）译码，同上，区块译码恢复的过程也分为2个阶段，如算法3所示。

算法3　区块级联译码

输入 节点n所属分片S，共识轮次e

输出 原始块B

1) 节点n在S内部发起译码请求并搜集共识轮次e的编码块EB^e

2) 遍历所有度值为1的EB^e

3) 执行LT译码操作并更新二分图直至不满足译码条件，得到$\mathrm{IB}=\left\{\mathrm{IB}_1^e,\cdots,\mathrm{IB}_k^e\right\}$

4) 结束遍历

5) 判断$\mathrm{hash}(\mathrm{IB}_i^e)==\mathrm{hb}(B_i^e)$，剔除恶意区块，得到原始块与校验块$\left\{B^e|C^e\right\}$

6) 如果译码结果包含所需原始块B

7) 返回原始块B并向客户端响应//乐观情况下直接由中间块得到原始块

8) 否则进入LDPC码译码阶段，对B^e与C^e执行译码操作得到$B=\left\{B_1^e,\cdots,B_k^e\right\}$

9) 判断$\mathrm{hash}(B_i^e)==\mathrm{hb}(B_i^e)$，剔除恶意区块，得到原始块$B^e$

10) 如果译码结果包含所需原始块B

11) 返回原始块B并向客户端响应结束函数

具体过程如下。

第一阶段。首先节点会向分片内其他节点进行广播请求其他编码块，在搜集到一定数量的编码块$\mathrm{EB}=\{\mathrm{EB}_1,\mathrm{EB}_2,\cdots,\mathrm{EB}_n\}$时，节点会开始对编码块进行LT译码操作，译码过程如下。

1) 通过编码块EB与中间块IB的对应信息生成二分图。

2) 找到二分图中度值为1的编码块EB，称之为单例，如果存在这样的编码块，将单例编码块的值赋予连接的中间块，删除单例编码块与中间块的连线，并删除该中间块与其他编码块的连线，更新二分图；若不存在此单例，则此次译码失败。

3) 对更新后的二分图重复上述操作，直到所有编码块被译码恢复或不满足译码条件。

在LT译码阶段结束后，根据中间块系统编码的特点，节点通过已恢复的中间块可以直接得到原始块，即可以通过IB的前k位与原始块B的关系$\mathrm{IB}=\{\mathrm{IB}_1,\mathrm{IB}_2,\cdots,\mathrm{IB}_k\}=\{B_1,B_2,\cdots,B_k\}$得到，同时将节点计算译码得到的原始块哈希值与全局的区块头部链中的哈希值进行比较，剔除恶意编码块生成的无效区块。若是该阶段恢复了k个原始块或其中包含用户请求的区块，则此次译码成功；否则，进入第二阶段。

第二阶段。通过第一阶段的LT译码，节点译码得到的中间块不足以恢复完整的k个原始区块或者其中不包含用户请求的区块。此时节点会对中间块IB进行LDPC译码，若所缺区块处于LDPC码的纠错能力范围之内，则可以得到所需区块B，第二阶段译码过程如下。

首先节点会得到不完整的原始块和中间块，通过对所缺原始块相关联的所有中间块进行异或操作来计算原始块B。之后节点计算B的哈希值并与存储的全局的区块头部链进行比较，若相等则说明获得了正确的原始块，否则将其丢弃。同时更新校验矩阵，将中间块与原始块对应位置零并重复之前的步骤。在该阶段结束后，若没有恢复k个原始区块，则此次译码失败，此时会向更多节点请求编码块，在已经恢复的原始块基础上继续译码找到所缺的区块。

5　性能分析

本节对本文方案从数据可用性、鲁棒性、通信开销、编译码复杂度等角度进行了分析，并就其中几个方面的性能与文献[14,16-17]方案进行了比较，如表2所示。

表 2 现有编码方案性能对比

方案	编码方案	数据可用性	通信开销	是否支持分片	编码复杂度	译码复杂度
文献[14]	LDPC 码	弱	高	不支持	$O(n)$	$O(n^2)$
文献[16]	RS 码	弱	高	不支持	$O(n^2)$	$O(n^2)$
文献[17]	LT 码	强	高	不支持	$O(\log(n))$	$O(n\log(n))$
本文方案	LT 码	强	低	支持	$O(n)$	$O(n)$

5.1 数据可用性

传统有率纠删码的编码率可定义为$\frac{k}{n}$，即通过k个输入符号获得$n-k$个编码冗余符号。在传统编码方案中，每个节点都必须按照既定的编码策略存储不同的编码块，当节点数量发生变化时，为保证区块链的数据可用性，节点需要对原始区块进行重新编译码并存储，这在节点频繁进出的区块链网络中会导致大量的通信开销。

本文方案采用的级联编码基于无码率纠删码，可动态调整编码率以适应不同的条件。对于k个输入符号，节点通过度分布函数能产生任意数量的编码符号并存储，而不需要按照编码策略保存固定的编码块，同时在节点动态进出的网络环境下，只要节点通过其余诚实节点请求到的编码块数量达到K（K略大于k），就能通过译码算法以极大概率恢复原始区块，因此本文方案具有较好的数据可用性。同时节点的可扩展性也在考虑范围之内，方案通过每轮共识结束前的特殊请求块来处理加入或者退出的节点。在特殊请求块时期，新增节点会向分片中其余节点请求编码块进行编译码并存储随机的编码块。而对于退出的节点，只有当分片内节点可请求的编码块数量小于阈值K时，分片中的节点才会对原始区块进行重新编码并存储，期间并不影响其余分片的运行，以此适应区块链中动态节点变更下的数据可用性和系统可扩展性。

5.2 鲁棒性

本文将恶意节点在区块链中的可能行为定义为以下 3 种：① 在面对其他节点的编码数据块请求时保持沉默；② 对于其他节点的编码数据块请求，发送经过篡改的编码数据块 EB；③ 任意选取度值计算编码数据块 EB，发送未篡改但不正确的编码数据块。

本文方案考虑分片S_i中的节点N_i^s准备恢复共识轮次e_i中的区块子链s_i^e，N_i^s会将向其他节点请求到的编码数据块记作 $\mathrm{EB}=\{\mathrm{EB}_1,\mathrm{EB}_2,\cdots,\mathrm{EB}_n\}$，只有收集到一定数量的编码数据块时节点才会开始尝试译码，因此面对沉默节点，节点会尝试联系额外的其他节点来获取足够多的编码数据块。除此之外，每个存储节点N_i^s会存储完整区块链账本的全局的区块头部链 hb，hb 相比完整区块链要小得多，占用的存储空间几乎可忽略不计，当节点对 EB 进行译码操作后会得到原始块B_j，通过比较B_j与全局的区块头部链中对应的区块B_i的哈希值是否相等来判断节点N_i^s是不是恶意的，通过这种方式节点能够保证经过译码算法后得到的原始区块的真实性。

5.3 通信开销

在应用编码方案的传统区块链网络中，一个或者一组区块会通过编码策略被分为k个数据片段并随机存储于不同地理环境的区块链节点中，而在以往的研究中编码方案并未考虑到区块链节点之间的时延差异对区块恢复过程带来的影响。假设在区块恢复场景中，D表示一个原始块的平均大小，k表示随机存储编码块的区块链节点数，R表示平均数据传输速率，L_{avg}表示未分片场景下节点之间的通信时延，L_{shard}表示分片后片内节点的通信时延。

在未分片和分片后的场景中，考虑到数据传输时延和通信时延，由以上假设可得未分片场景中节点恢复一个区块所需的时间为

$$T=\frac{D}{kR}+2L_{\mathrm{avg}} \tag{2}$$

在分片场景中，节点恢复一个区块所需的时间为

$$\Delta T=\frac{D}{kR}+2L_{\mathrm{shard}} \tag{3}$$

在未分片的区块链网络中，编码片段存储于不同地理位置的节点中，在区块恢复过程中会接入多种网络环境，导致数据传输时间不确定，特别是在与远程节点进行交互时会导致较高的时延，从而增加数据损坏的风险导致数据重传。而分片技术引入后将区块的恢复限定在一组时延较低的分片中，优化了分片内节点之间的通信环境，缩短了数据的传输时间，减少了数据重传需求，降低了区块恢复带来的整体通信时延。由式(3)可知，节点经过分片后$\Delta T \ll T$。

5.4 编译码复杂度

用$(k,\mathrm{LDPC},\Omega_d)$表示本文方案，其中，$k$表示作为输入的一组原始区块，LDPC 为该方案采用的预编码，Ω_d是 LT 编码采用的度分布函数。本文方案的编译码开销分为两部分，用$\varepsilon_{\mathrm{pre}}$表示 LDPC 码的编译码开销，$\varepsilon_{\mathrm{LT}}$表示 LT 码的编译码开销，两者满足如下关系

$$(1+\varepsilon)=(1+\varepsilon_{\text{pre}})(1+\varepsilon_{\text{LT}}) \tag{4}$$

由于文献[18]建议$\varepsilon_{\text{LT}}=\frac{\varepsilon}{2}$，于是总编码开销与预编码开销的关系为

$$\varepsilon=\frac{2\varepsilon_{\text{pre}}}{1-\varepsilon_{\text{pre}}} \tag{5}$$

由式(5)可知，预编码码率$R=(1+\varepsilon)\left(1+\frac{\varepsilon}{2}\right)$，译码复杂度为$O\left(\frac{k\log\left(\frac{1}{\varepsilon}\right)}{R}\right)$。

由于本文方案采用BP译码方式对编码块EB进行译码，编译码的复杂度等于对二分图中边的数量的操作。因此方案的编译码复杂度为$E(\text{pre})+\overline{d}k\frac{O(1+\varepsilon)}{R}$，其中，$E(\text{pre})$表示预编码的代数运算次数，$\overline{d}$为LT编码的平均度值，可以表示为

$$\begin{aligned}&\overline{d}=\frac{\mu}{\mu+1}+\frac{1}{\mu+1}\sum_{i=1}^{D-1}\frac{1}{i}+\frac{D+1}{(\mu+1)D}=\\&1+\frac{1}{\mu+1}\sum_{i=1}^{D}\frac{1}{i}=\\&\log\left(\frac{1}{\varepsilon}\right)+\alpha+O(\varepsilon)=O\left(\log\left(\frac{1}{\varepsilon}\right)\right)\end{aligned} \tag{6}$$

其中，$1<\alpha<1+c$，c为欧拉常数。因此本文方案的总编译码复杂度可以表示为

$$\begin{aligned}&E(\text{pre})+\overline{d}k\frac{O(1+\varepsilon)}{R}=\\&O\left(k\frac{\log\left(\frac{1}{\varepsilon}\right)}{R}\right)+O\left(k\frac{\log\left(\frac{1}{\varepsilon}\right)(1+\varepsilon)}{R}\right)=\\&O\left(k\log\left(\frac{1}{\varepsilon}\right)\right)\end{aligned} \tag{7}$$

由式(7)可知，对比文献[17]，本文方案的译码复杂度由$O(k\log(k))$降至$O(k)$，其编译码的计算开销与原始区块长度k成线性关系。

6　实验结果

6.1　实验设置

为了对方案性能进行有效评估，本文在开源的区块链平台Tendermint上进行测试。Tendermint提供了可插拔的共识引擎，可用不同的编程语言来实现拜占庭共识，同时通过应用程序接口可以自定义区块链的状态机，与不同的数据库进行集成，实现处理交易与存储数据的功能，是目前比较主流的测试框架[26]。

实验使用Go语言修改Tendermint的数据存储方式与配置文件，实现本文方案的编码存储策略。所有实验在5台机器组成的集群上完成，每台机器最多运行8个节点实例，以组成一个40个节点规模的区块链网络，节点之间通过传输控制协议（TCP, transmission control protocol）进行通信以保持对等连接。本文方案的机器硬件配置如表3所示。实验将文献[17]方案作为本文方案的对比，为每个节点分配1 GB的存储空间，将区块的大小定义为0.1 MB。为了模拟拜占庭环境，方案定义拜占庭节点在区块译码过程中可能会拒绝响应或者发送经过篡改的信息。

表3　机器硬件配置

机器编号	处理器	内存/GB
1	Intel Core i5-9500	8
2	Intel Core i9-12900H	16
3	Intel Core i5-12500H	16
4	Intel Core i7-10750H	8
5	Intel Core i5-11300H	16

6.2　译码成本

在基于纠删码的区块链存储方案中，为了能成功恢复长度为k的原始区块，节点至少需要请求K（$K>k$）个编码块并对其进行译码操作，K值代表方案的译码成本，同时K值的大小也是衡量方案性能的重要指标。

为了评估本文方案性能与资源消耗之间的关系，实验以译码成本作为衡量指标，通过在拜占庭环境下比较恢复不同长度的原始区块所需的译码成本来分析方案的性能，如图8所示，并在后续通过存储开销、通信开销和译码开销3个维度衡量本文方案的性能。

从图8中可以看出，随着需要恢复的原始区块长度的增加，节点的译码成本也在增加，但两者始终保持在一定比例，这是因为本文方案通过添加预编码层在编码过程中实现了对原始区块的良好覆盖，使得节点在恢复不同长度的原始区块时都保持着良好的译码成本。而随着节点的拜占庭比例升高，节点的译码成本与原始区块长度的比例分别维持在130%、140%和155%，这是因为恶意节点可能会发送无效或者篡改过的编码块，导致译码节点需要联系更多的诚实节点获取编码块以成功恢复原始区块。

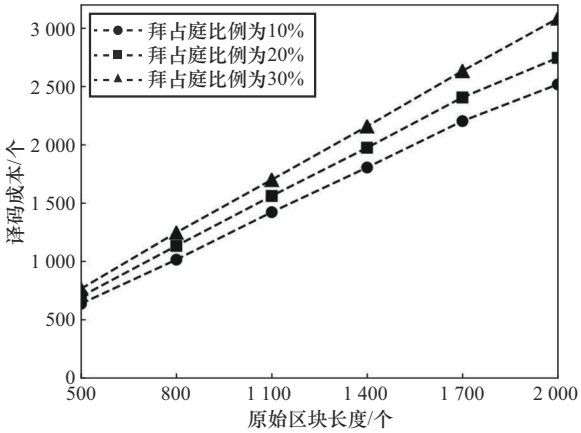

图8 区块长度对译码成本的影响

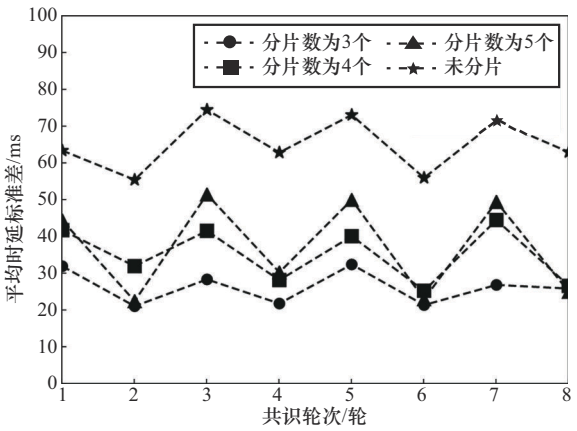

图9 平均时延标准差随共识轮次的变化

6.3 分片性能测试

为了评估Metis算法在区块链动态网络拓扑（即节点的加入和退出）中的性能，实验初始化构建了一个由40个节点组成的区块链网络，这些节点按照Metis算法分配至不同分片中。为模拟节点的加入和退出，系统将随着时间添加或移除节点，并测试各分片中节点共识一轮的通信时延。

实验以共识轮数为时间单位，在第1轮共识前移除10个节点，在第2轮共识前添加10个节点，以此类推，并以各分片中节点共识一轮通信的平均时延标准差作为衡量指标，测试在不同分片数量下本文方案在区块链动态网络环境下的性能。

从图9中可以看出，经过分片后节点的平均时延标准差远低于未分片节点的平均时延标准差，证明了Metis算法对优化节点通信时延的有效性。而在分片的情况下，分片数越多，平均时延标准差的波动也越大。这是因为分片数越多，每个分片内的节点就越少，节点之间的通信时延受到动态网络环境中节点进出的影响就越大。在分片数为5个的情况下，节点的平均时延标准差波动范围为20~50 ms，依旧优于未分片条件下方案的平均时延标准差。

6.4 存储开销

为了评估方案的存储开销，实验添加未使用纠删码的区块链存储策略f-replica作为对比。f-replica策略为了保证系统中至少有一个诚实节点会保存完整的区块，会将每个区块存储在至少$f+1$个节点上，其中f表示恶意节点的最大数量。实验以平均区块存储消耗为衡量指标，即区块链的总存储消耗除以原始区块的数量。

从图10中可以看出，对于f-replica存储策略，每个区块至少会存储在$\left\lfloor \frac{N-1}{3} \right\rfloor + 1$个节点中，导致f-replica策略下单区块的存储消耗会随节点数量的增加而线性增加。文献[17]方案与本文方案的平均区块存储消耗会随节点数量的增加而在极小区间内波动，这是因为与区块链传统的多副本存储策略相比，纠删码通过极少的存储冗余实现了相同的数据可用性与可靠性，大幅降低了区块链系统的存储开销。

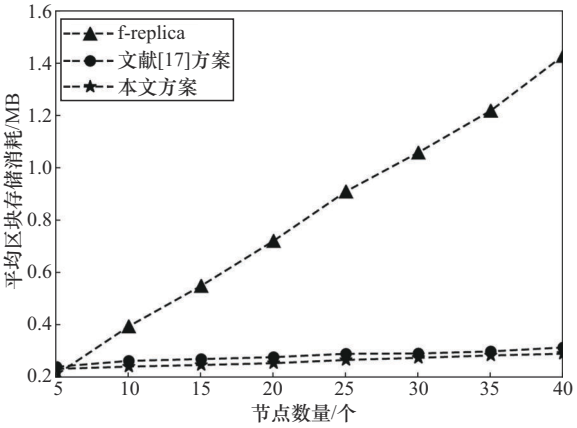

图10 节点数量对平均区块存储消耗的影响

6.5 通信开销

为了评估在区块译码过程中节点在请求编码块时带来的通信开销，实验将节点规模设置为16~40个，分片数量设置为4个，以恢复单组原始区块的平均通信时延作为衡量指标，即节点从发起恢复一组原始区块的请求到接收到足够多的编码块的时间。系统会随机在不同分片中发起原始区块的恢复请求，最后计算所有请求的平均通信时延。

从图11中可以看出，文献[17]方案中的节点并

未进行分片，节点在原始区块恢复请求中会随机向其他节点发起编码块的传输请求，导致发起一组区块恢复所需的通信开销较大，随着区块链节点规模的扩大，恢复单组原始区块的平均通信时延呈逐渐上升的趋势。本文方案通过Metis算法划分了节点归属并优化了分片内通信开销，因此发起一组原始区块恢复所需的平均通信时延相较文献[17]方案大幅降低，并且随着节点规模的扩大，片内节点发起一组原始区块恢复请求的平均通信时延上升幅度也低于文献[17]方案。

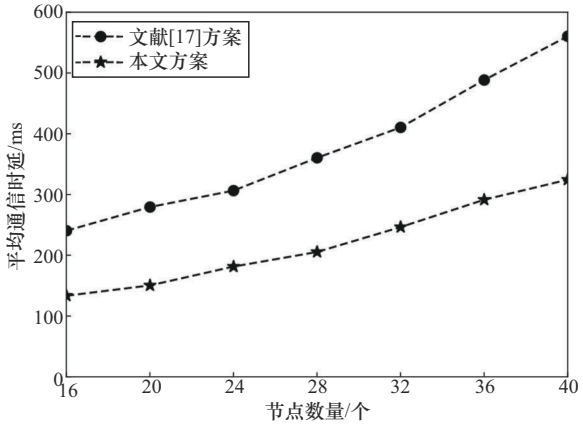

图11　节点数量对平均通信时延的影响

6.6　译码开销

本文方案的译码开销包含节点向分片内的其他节点请求编码块所带来的通信开销与节点进行译码时根据译码算法带来的计算开销。为了综合评估本文方案的译码开销，实验在6.4节的基础上，测试了节点从发起一组原始区块的恢复请求到完成译码的时间。实验以单组原始区块的译码时延作为衡量指标，即节点成功恢复一组原始区块所消耗的时间，在原始区块500~2 000的长度以及没有拜占庭节点的条件下分别对2种方案进行了测试，结果如图12所示。为了得到在拜占庭环境中节点的译码时延，实验在原始区块长度为1 000个的条件下，分别对2种方案在拜占庭节点数量0~10个的情况下进行了测试，如图13所示。

综上所述，在理想情况下（不存在拜占庭节点），文献[17]方案的译码开销随着所译码区块长度的增加而呈对数级增长，在原始区块长度达到2 000个时译码开销达到了60多秒，这是因为LT码在编译较长的区块链时需要进行更多的异或运算。而本文方案所需的译码时延随着区块长度的增加呈线性增长，当所译区块长度较大时，本文方案的译码开销远小于文献[17]方案。而在拜占庭环境中，本文方案的译码开销在固定原始区块长度下相比文献[17]方案也降低了约15%。

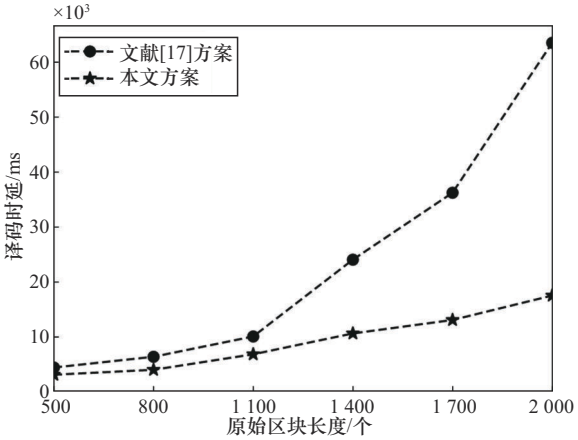

图12　区块长度对区块译码时延的影响

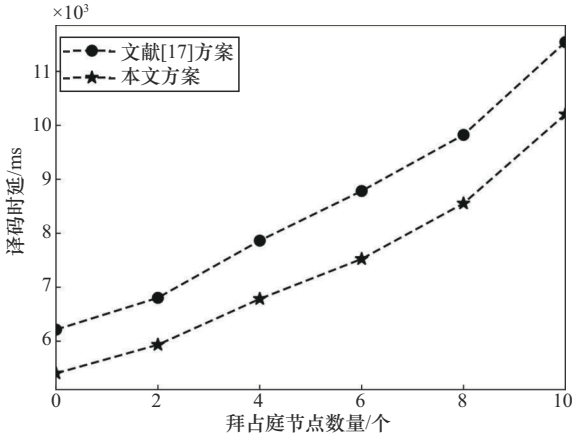

图13　拜占庭节点数量对译码时延的影响

7　结束语

为了解决纠删码应用于区块链存储中带来的计算开销与通信开销过高的问题，本文提出了基于级联编码的区块链分片存储方案。该方案通过添加预编码层改进了现有无码率纠删码，降低了译码过程中节点的计算开销，并基于节点间的时延感知设计了通信负载均衡的区块链分片方法，降低了译码过程中的通信开销性能。通过实验结果可知，相比于传统区块链编码存储方案，本文方案具有更优的计算与通信开销性能。在未来工作中，笔者将研究本文方案在不同区块链中的适用性并探索更佳的优化策略来提高纠删码在区块链中的性能。

参考文献：

[1] 孙知信, 张鑫, 相峰, 等. 区块链存储可扩展性研究进展[J]. 软件学报, 2021, 32(1): 1-20.
SUN Z X, ZHANG X, XIANG F, et al. Survey of storage scalability on blockchain[J]. Journal of Software, 2021, 32(1): 1-20.

[2] 田有亮, 袁延森, 高鸿峰, 等. 基于激励相容的权益分散共识算法[J]. 通信学报, 2022, 43(12): 101-112.
TIAN Y L, YUAN Y S, GAO H F, et al. Equity decentralized consensus algorithm based on incentive compatibility[J]. Journal on Communications, 2022, 43(12): 101-112.

[3] FAN X, NIU B N, LIU Z L. Scalable blockchain storage systems: research progress and models[J]. Computing, 2022, 104(6): 1497-1524.

[4] MATZUTT R, KALDE B, PENNEKAMP J, et al. CoinPrune: shrinking Bitcoin's blockchain retrospectively[J]. IEEE Transactions on Network and Service Management, 2021, 18(3): 3064-3078.

[5] KHAN D, JUNG L T, HASHMANI M A. Systematic literature review of challenges in blockchain scalability[J]. Applied Sciences, 2021, 11(20): 9372.

[6] MITRA D, DOLECEK L. Patterned erasure correcting codes for low storage-overhead blockchain systems[C]//Proceedings of the 2019 53rd Asilomar Conference on Signals, Systems, and Computers. Piscataway: IEEE Press, 2019: 1734-1738.

[7] HE G B, SU W, GAO S. Chameleon: a scalable and adaptive permissioned blockchain architecture[C]//Proceedings of the 2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN). Piscataway: IEEE Press, 2018: 87-93.

[8] SHAH M, SHAIKH M, MISHRA V, et al. Decentralized cloud storage using blockchain[C]//Proceedings of the 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184). Piscataway: IEEE Press, 2020: 384-389.

[9] GOINT M, BERTELLE C, DUVALLET C. Secure access control to data in off-chain storage in blockchain-based consent systems[J]. Mathematics, 2023, 11(7): 1592-1601.

[10] NAGAYAMA R, BANNO R, SHUDO K. Trail: a blockchain architecture for light nodes[C]//Proceedings of the 2020 IEEE Symposium on Computers and Communications (ISCC). Piscataway: IEEE Press, 2020: 1-7.

[11] BÜNZ B, KIFFER L, LUU L, et al. FlyClient: super-light clients for cryptocurrencies[C]//Proceedings of the 2020 IEEE Symposium on Security and Privacy (SP). Piscataway: IEEE Press, 2020: 928-946.

[12] PERARD D, LACAN J, BACHY Y, et al. Erasure code-based low storage blockchain node[C]//Proceedings of the 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData). Piscataway: IEEE Press, 2018: 1622-1627.

[13] WANG Z Z, WANG H X, SHAO A R, et al. An adaptive erasure-coded storage scheme with an efficient code-switching algorithm[C]//Proceedings of the 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS). Piscataway: IEEE Press, 2020: 1177-1178.

[14] WU H H, ASHIKHMIN A, WANG X D, et al. Distributed error correction coding scheme for low storage blockchain systems[J]. IEEE Internet of Things Journal, 2020, 7(8): 7054-7071.

[15] RAMAN R K, VARSHNEY L R. Dynamic distributed storage for blockchains[C]//Proceedings of the 2018 IEEE International Symposium on Information Theory (ISIT). Piscataway: IEEE Press, 2018: 2619-2623.

[16] QI X D, ZHANG Z, JIN C Q, et al. A reliable storage partition for permissioned blockchain[J]. IEEE Transactions on Knowledge and Data Engineering, 2021, 33(1): 14-27.

[17] KADHE S, CHUNG J, RAMCHANDRAN K. SeF: a secure fountain architecture for slashing storage costs in blockchains[J]. arXiv Preprint, arXiv:1906.12140, 2019.

[18] LUBY M. LT codes[C]//Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings. Piscataway: IEEE Press, 2002: 271-280.

[19] MACKAY D J C. Fountain codes[J]. IEE Proceedings Communications, 2005, 152(6): 1062-1068.

[20] ZAMANI M, MOVAHEDI M, RAYKOVA M. RapidChain: scaling blockchain via full sharding[C]//Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2018: 931-948.

[21] LUU L, NARAYANAN V, ZHENG C D, et al. A secure sharding protocol for open blockchains[C]//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2016: 17-30.

[22] WANG P, CHEN W Q, LIN S L, et al. Consensus algorithm based on verifiable quantum random numbers[J]. International Journal of Intelligent Systems, 2022, 37(10): 6857-6876.

[23] HUANG H W, PENG X W, ZHAN J Z, et al. BrokerChain: a cross-shard blockchain protocol for account/balance-based state sharding[C]//Proceedings of the IEEE Conference on Computer Communications. Piscataway: IEEE Press, 2022: 1968-1977.

[24] WANG J, HAN C C, YU X F, et al. Distributed secure storage scheme based on sharding blockchain[J]. Computers, Materials & Continua, 2022, 70(3): 4485-4502.

[25] SHOKROLLAHI A. Raptor codes[J]. IEEE Transactions on Information Theory, 2006, 52(6): 2551-2567.

[26] KWON J, BUCHMAN E. Cosmos whitepaper[J]. A Network of Distributed Ledgers, 2019, 27: 1-32.

[作者简介]

田有亮（1982– ），男，贵州盘县人，博士，贵州大学教授、博士生导师，主要研究方向为算法博弈论、密码学与安全协议、大数据安全与隐私保护等。

黄钰清（1997– ），男，贵州瓮安人，贵州大学硕士生，主要研究方向为区块链技术、共识算法等。

王帅（1998– ），男，贵州毕节人，贵州大学博士生，主要研究方向为人工智能、机器学习等。