基于纠删码的区块链存储优化

樊玉琦[1),2)] 盛 东[2)] 王伦飞[2)]

[1)]（大数据知识工程教育部重点实验室(合肥工业大学) 合肥 230601）

[2)]（合肥工业大学计算机与信息学院 合肥 230601）

摘 要 区块链具有去中心化、不可篡改、可追溯以及公开透明等特性，可以解决去中心化网络中节点之间相互不信任的问题，为构建价值互联平台提供了可能.然而，区块链要求每个节点都存储一份完整的数据，以高存储冗余来保证数据的可靠性，给节点带来了巨大的存储压力，降低了存储资源的利用效率，也导致系统的存储可扩展性成为区块链性能的一个瓶颈.采用纠删码来编码存储在区块链中的数据可以有效地减少存储冗余，但存储冗余的减少会降低数据的可靠性，引发数据的重组消耗，提高数据的读取延迟.目前已有研究在区块链编码数据块的存储分配阶段并没有考虑节点间延迟、区块存储位置等因素对数据可靠性和读取延迟的影响.本文在基于纠删码的BFT联盟链中，研究编码数据块的存储数量及存储位置决策问题，以在满足数据可靠性的约束下实现数据存储代价和数据读取性能的平衡.针对编码数据块的存储数量及存储位置决策问题，本文提出了延迟感知的编码数据块分配算法(Latency-aware Encoded data chunks Allocation algorithm, LEA).算法LEA首先求解编码数据块的存储数量及存储位置决策问题的松弛问题以及该松弛问题的对偶问题，然后根据松弛问题及其对偶问题的最优解依次为每个编码数据块确定其存储数量和存储位置，最后调整得到的编码数据块存储分配方案使其满足被松弛的约束条件.理论分析证明，算法LEA是$\ln 3+2$近似算法.仿真环境和真实联盟链系统中的实验结果表明，算法LEA可以有效降低区块链系统的存储冗余，提高系统的存储可扩展性，并实现良好的数据存储代价和数据读取性能的平衡.

关键词 区块链；存储优化；延迟；纠删码；可靠性

中图法分类号 TP393 **DOI号** 10.11897/SP.J.1016.2022.00858

Blockchain Storage Optimization Based on Erasure Code

FAN Yu-Qi[1),2)] SHENG Dong[2)] WANG Lun-Fei[2)]

[1)](*Key Laboratory of Knowledge Engineering with Big Data (Hefei University of Technology), Ministry of Education, Hefei* 230601)

[2)](*School of Computer Science and Information Engineering, Hefei University of Technology, Hefei* 230601)

Abstract Blockchain has the characteristics of decentralization, unforgeability, traceability, transparency, etc. Therefore, blockchain can solve the problem of mutual distrust between nodes in the decentralized network and provide a viable way to build a value-connected platform. However, blockchain requires each node to store a complete copy of data to ensure data reliability with high storage redundancy. Unfortunately, it also brings huge storage pressure to blockchain nodes and reduces the utilization of storage resources, which makes the storage scalability be a bottleneck of blockchain. Meanwhile, with the increasing number of transactions, the size of blockchain also increases rapidly, which hinders the nodes with limited storage capacity to join the blockchain system and hence weakens the decentralization of the system. Using erasure code to

收稿日期：2021-07-02；在线发布日期：2021-11-05. 本课题得到国家重点研发计划项目(2018YFB2000505)、国家自然科学基金(61806067)、安徽省重点研发计划项目(201904a06020024)资助. **樊玉琦**(通信作者)，博士，副教授，中国计算机学会(CCF)会员，主要研究领域为区块链、计算机网络、云计算、人工智能等. E-mail：yuqi.fan@hfut.edu.cn. **盛 东**，硕士研究生，中国计算机学会(CCF)会员，主要研究领域为区块链. **王伦飞**，硕士研究生，主要研究领域为区块链、计算机网络.

encode the data stored in the blockchain can effectively reduce the storage redundancy. Nevertheless, the reduction of storage redundancy will reduce the reliability of the data, incur data recovery costs, and increase the data access delay. The existing work on reducing the storage redundancy in blockchains ignores the impact of inter-node delay and encoded data chunk storage locations on data reliability and data access delay during the data chunk placements, while there is a tradeoff between data storage cost and data access latency. In this paper, we study the decision problem of the number and locations of encoded data chunk copies with the erasure code in BFT permissioned blockchains, with the aim to achieve a good balance between data storage cost and data access latency under the data reliability constraint. We also propose a Latency-aware Encoded data chunks Allocation algorithm (LEA). Firstly, algorithm LEA solves the relaxation problem of the studied problem and the dual problem of the relaxation problem. The relaxation simplifies the problem model and facilitates the solution of the problem. We classify the solutions of the relaxation problem into clusters. Specifically, we select a node as the cluster center based on the optimal solution of the dual problem, and put the nodes into different clusters according to the optimal solution of the relaxation problem. Secondly, at least one node in each cluster is randomly selected to store the copy of the coded data chunk according to the optimal solution of the relaxation problem. Finally, the algorithm adjusts the storage allocation solution to satisfy the constraints of the decision problem. Theoretical analysis proves that algorithm LEA is a $\ln 3+2$ approximation algorithm to the data allocation decision problem. We conduct extensive experiments in both the simulation environment and the real permissioned blockchain system. The experimental results show that algorithm LEA can achieve good performance in terms of the tradeoff between data storage cost and data access delay, while satisfying the data reliability constraint. Algorithm LEA can effectively reduce the blockchain storage cost, enhance the blockchain storage capacity with the increase of the number of nodes, and realize the horizontal scalability of the system storage. Compared with the existing algorithms, algorithm LEA can achieve better data access performance in different networks, and the performance improvement can reach 15.2%. In addition, we evaluate the impact of the number of malicious nodes on the performance of the algorithm. When there are malicious nodes in the network, algorithm LEA can still obtain favorable data access performance while ensuring data reliability, which outperforms the existing algorithms by 15.4%.

Keywords blockchain; storage optimization; latency; erasure code; reliability

1 引 言

2008年，中本聪提出了一个不依赖于系统参与者之间信任的电子交易系统，并首次解释了比特币和区块链技术的原理[1]. 区块链具有去中心化、不可篡改、可追溯以及公开透明等特征，基于这些特征，区块链解决了去中心化网络中节点之间相互不信任的问题，确保了交易的可靠性和安全性[2-3]. 目前，区块链在加密货币、信息安全、物联网以及公共社会服务等领域都得到了广泛的应用[4]. 例如在加密货币领域，截至2021年4月，加密货币市值统计网站CoinMarketCap①显示，全球共有9192种加密货币，总市值超过2万亿美元，其中比特币和以太币市值分别约占54.6%和12%；而在信息安全领域，如身份认证和访问控制都在尝试结合区块链技术[5-6]. 预计到2025年，区块链企业年收入将增加到约200亿美元[7].

存储可扩展性是影响区块链发展的关键因素. 区块链要求每个节点都存储一份完整的数据，在系统中部分节点的数据发生丢失或者被恶意篡改的情况下，其余节点依然拥有正确且完整的数据，保证了数据的可靠性，提高了系统的可用性[8]；但另一方面，每个节点都需要消耗大量的存储资源来保存冗余数据，并且需要不断地同步最新数据，导致整个系

① CoinMarketCap. https://coinmarketcap.com

统的存储能力不会随着节点数量的增加而增强，而是取决于系统中的瓶颈节点[9]. 冗余存储的方式不仅降低了存储资源的利用效率，而且阻碍了存储能力有限的普通节点加入系统，进而限制了区块链的应用和发展[10]. 特别的，随着交易数量的不断增加，区块链的存储需求正以接近指数级的速度增长，很容易超过存储设备的摩尔定律[11]. 截至2021年4月，一条完整的比特币区块链的大小已经达到338 GB①；截至2020年11月，高吞吐量区块链Ripple的大小已经达到14 TB，并且以每天12 GB的速度增长②. 这些数据说明区块链中的节点需要付出很高的存储代价来保存所有的数据，并且需要不断地增加存储空间以满足快速增长的存储需求. 由此可以看出，存储可扩展是区块链中一个亟待解决的重要问题.

公有链主要应用于数字货币、证券交易等金融领域[12]，首要目标是保证数据的安全和不可篡改，其共识速度慢，产生数据的速度较慢[13]. 联盟链只针对某个特定群体的成员和有限的第三方，常采用PBFT等协议来实现更高的事务吞吐量，其产生数据的速度更快. 由于快速增长的数据量很容易突破单个节点的存储容量上限，而传统的全复制方案（每个节点都存储一份完整数据）需要消耗大量的存储资源，进而导致高昂的系统构建和维护成本，因此在联盟链中如何解决存储可扩展性问题也显得更加迫切[14].

在数据库等领域中，采用纠删码能够有效减少存储系统的存储开销，因此已有研究开始将纠删码应用到区块链中，通过纠删码来减少区块链的数据存储冗余[14]. RS码（Reed-Solomon Code）是由Reed和Solomon在1960年提出的纠删码，目前被广泛使用[15]. 为区块链设计一个基于纠删码的存储方案是可行的. 例如，在一个有$3f+1$个节点的区块链系统中，如果采用PBFT作为其共识协议，那么整个系统中至少会存在$2f+1$个非故障节点[16]. 先将系统中的原始数据块划分为多个组，每组$2f+1$个块，然后通过RS码将每组原始数据块编码成$3f+1$个编码数据块，每次编码都会额外生成f个校验块，最后将$3f+1$个编码数据块分布在$3f+1$个节点上. 因为至少有$2f+1$个编码数据块存储在非故障节点上，并且RS码可以在丢失不超过f个编码数据块的情况下恢复原始的$2f+1$个数据块，所以每个原始数据块不需要存储在所有的节点中，也即每个块的存储消耗从$O(n)$减少到$O(1)$.

然而，数据存储冗余的降低也会引发新的问题，包括数据安全性降低和数据读取延迟增高等问题. 数据读取性能是区块链的一个重要性能，因为读取延迟会影响系统的共识等过程. 同时，数据存储代价与数据读取性能之间存在折中关系. 因此在利用纠删码减少区块链存储冗余时，面临以下两个挑战：(1)如何保证编码数据块丢失数量不会超过校验块数量；(2)如何平衡数据存储代价与数据读取性能.

本文在基于纠删码的BFT联盟链中，研究编码数据块的存储数量及存储位置决策问题，以在满足数据可靠性的约束下获得平衡的数据存储和读取性能. 本文的主要贡献如下：

(1) 在BFT联盟链中，提出了基于纠删码的数据编码方案，该方案先对系统中的原始区块进行分组，然后对每组原始区块使用RS码编码生成相应的校验块，最后将原始区块和校验块分配到不同的节点上存储. 该方案保证在发生数据块丢失时通过纠删码可以解码恢复出丢失的数据块，从而避免了每个节点均存储一份完整数据的弊端，减少了数据的存储冗余.

(2) 研究了编码数据块的存储数量及存储位置决策问题，提出了延迟感知的编码数据块分配算法（Latency-aware Encoded data chunks Allocation algorithm，LEA），以在满足数据可靠性的约束下获得平衡的数据存储和读取性能. 算法LEA首先求解编码数据块的存储数量及存储位置决策问题的松弛问题以及该松弛问题的对偶问题，然后根据松弛问题及其对偶问题的最优解依次为每个编码数据块确定其存储数量和存储位置，最后调整得到的编码数据块存储分配方案使其满足被松弛的约束条件. 理论分析证明，算法LEA是$\ln 3+2$近似算法.

(3) 本文在仿真环境和真实的联盟链系统中都进行了实验，实验结果表明算法LEA可以有效降低区块链系统的存储开销，提高系统的存储可扩展性，并实现良好的数据存储代价和数据读取性能的平衡.

本文的组织结构如下：第一节介绍了基于纠删码的区块链存储优化的背景及研究意义；第二节介绍了相关工作；第三节介绍了系统模型，并给出了编码数据块的存储数量及存储位置决策问题的定义；第四节详细描述了延迟感知的编码数据块分配算法

① Blockchain Size. https://www.blockchain.com/charts/blocks-size

② Ripple Documentation. https://xrpl.org/capacity-planning.html

LEA，并证明了算法LEA是ln 3＋2近似算法；第五节在仿真环境和真实的联盟链系统中进行实验验证；第六节对本文进行了总结.

2 相关工作

随着交易数量的不断增加，区块链的大小也随之迅速增加，使得区块链节点需要消耗大量的存储资源，导致区块链网络中愿意充当全节点的节点数量减少，从而影响了区块链的去中心化程度和可靠性.事实上，自比特币诞生后不久，为了降低节点的存储消耗，研究者已经提出了两种比较直接的方法，即删除节点中旧的交易历史记录而只保留较新的数据[1]，及在系统中部署轻量级节点只保存区块头信息而不保存区块体，如Zcash①、Byteball②等.但这两种方法都是以丢失数据为代价来减少节点所需的存储空间，会降低数据的可靠性和系统的可用性[17]，而系统的存储能力并未得到有效的提升.

目前，区块链中解决存储可扩展性问题的方案主要分为两类：链下存储和链上存储[8].链下存储是指将区块链中的数据从区块链中转移到链下系统进行存储，而只在区块链的区块中保存指向链下数据的指针；链下存储主要有基于分布式哈希表(Distributed Hash Table，DHT)[18]、基于星际文件系统(Inter Planetary File System，IPFS)[19-20]以及基于云[21]等解决方案.Zyskind等人[18]结合区块链和DHT设计了一种链下存储模式，将原始数据保存在链下的DHT中，而区块链中的节点只存储原始数据的引用.Xu等人[19]使用智能合约和IPFS在以太坊上实现了去中心化的社交网络应用，用户数据保存在以太坊上，而IPFS则用于保存大型文件数据.Ali等人[20]针对物联网的数据隐私问题提出了一种结合区块链和IPFS的网络架构，将物联网数据分组并存储在IPFS中，而区块链本身只保存IPFS文件的散列，以减少节点所需的存储空间.He等人[21]提出了一种可扩展和自适应的区块链体系结构，引入了一种与云基础设施交互的协作机制来解决存储可扩展性问题，将最近一段时间生成的数据保存在区块链中，而以前的数据则存储到云上.上述三种链下存储方案虽然可以减轻区块链节点的存储压力，但是会引入一些新的问题.基于DHT和基于IPFS的方案都需要先保证链下存储系统的安全可靠，从而才能保证区块链系统中数据的可用性[8]；而基于云的方案会引入中心化的机构(云服务提供商)，此外，区块链节点很难参与到云存储中，这严重削弱了区块链的去中心化程度[21].

链上存储需要每个节点根据预定的规则存储对应的部分数据，不需要存储完整的数据，通常需要区块链中的节点进行协作；链上存储常见的主要有基于编码[22]、基于集群[23]和基于分片[24-25]共三种方法.Abe等人[23]提出了一种分布式存储方案，使用比特币网络中的节点子集构建DHT集群，集群中的节点只会保存区块数据的一部分，而完整的区块链数据则由同一个集群中的节点共同维护.Zamani等人[24]提出了一种基于分片的公有链协议，将网络中的节点划分成多个较小的称作委员会的节点子集，不同的委员会维护不相交的账本子集以实现交易的并行处理，该协议通过将完整的账本数据分布到不同的委员会中，以减少节点所需的存储空间.Wang等人[25]设计了一个基于区块链的共识系统，通过运行多个并行且独立的单链系统，使得节点只需存储和所在单链系统相关的交易，减少了节点的数据存储冗余.基于集群和基于分片的方案都是将整个系统的存储压力分摊到不同的节点子集中，这种方式虽然可以提高存储资源的利用效率，但是会消耗系统的带宽资源.在基于集群的方法中，不同集群中的节点在读取数据时会产生较高的消息传递开销[23]；而在基于分片的方案中，节点在验证数据时可能会涉及多个分片，产生较高的通信开销[24-25].

针对基于编码的链上存储方案，已有研究将纠删码应用到区块链的数据存储中，通过纠删码来减少区块链中存储的数据量.Perard等人[26]提出了利用纠删码来编码区块链数据的设想，将每个区块划分成多个片段，并对这些片段进行编码；另外，作者还引入了一种新的介于全节点和轻节点之间的低存储节点，该节点只存储区块链中每个区块的部分编码片段，在减少节点所需存储容量的同时，依然保留了区块链去中心化的特性.Dai等人[22]提出了一种类似的基于纠删码的编码机制以减少区块链的数据存储冗余.Wu等人[27]同样提出了利用纠删码来减少区块链节点存储需求的编码方案，该编码方案对区块链中的多个区块数据进行编码；此外，作者以LDPC码为例说明了编码方案可以显著减少每个节点所需的存储空间.Raman等人[28]提出了一种基于纠删码的分布式编码方案，该方案结合密码共享、私

① Zcash. https://zcash.readthedocs.io/en/latest
② ByteBall. https://byteball.org/Byteball.pdf

钥加密和分布式存储，使得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. Mitra等人[17]定义了一系列的节点失效模式，例如节点1和节点2同时失效或者节点3失效等，针对这一系列的节点失效模式，作者设计了一种基于纠删码的编码方案来对存储需求进行优化；该方案保证当失效模式属于预先定义的一系列节点失效模式时，区块链数据可以安全地进行恢复，但当失效模式不属于预先定义的一系列节点失效模式时，则不能保证区块链数据的可靠性. Zhao等人[29]提出了一种基于纠删码的区块链文件存储模型，并将其应用于超级账本区块链系统. Kadhe等人[11]提出了一种基于喷泉码（一种擦除码）的安全喷泉体系结构（Secure Fountain，SeF），通过该体系结构，全节点可以将经过验证的块编码成少量的编码块，从而降低其存储成本. Pal[30]也提出了一种使用喷泉码来编码区块链数据的方案. Qi等人[14]将纠删码应用于基于拜占庭容错（BFT）的联盟链中，同时为了提高数据读取性能而采用了多副本机制. Qi等人还在文献[31]中验证了一种基于联盟链的区块存储策略f-replica，每个区块至少存储在$f+1$个节点上，其中f是联盟链中恶意节点的最大数量，该策略可以确保系统中至少有一个诚实节点会保存完整的区块.

已有的解决区块链存储可扩展性问题的方案通常都会降低数据的可靠性. 在链下存储的方案中，需要将区块链上的数据转移至链下可信的分布式存储系统中，该方案虽可以减少区块链节点的存储需求，但是需要防止链下存储系统中的节点发生故障或者恶意篡改数据. 而在链上存储的方案中，每个节点根据预定的规则只需存储部分数据，存储资源的利用率得到了明显的提高，但是，由于恶意节点的存在，数据很有可能会被存储在恶意节点上，因而这种方案也很容易导致数据的篡改和丢失. 由上述分析可知，无论是在链下存储还是在链上存储方案中，恶意节点的存在都会降低数据的可靠性，并进一步的削弱区块链系统的整体可用性.

现有研究表明采用纠删码是减少区块链数据冗余的一种有效方法. 但是，使用纠删码会降低数据的可靠性，引发数据的重组消耗，增大数据的读取延迟[32-33]. 目前已有研究在区块链编码数据块的存储分配阶段并没有考虑节点间延迟、区块存储位置等因素对数据可靠性和读取延迟的影响. 而数据读取性能是区块链系统中的一种重要性能，因为读取延迟影响系统的共识等过程. 本文在基于纠删码的BFT联盟链中，研究编码数据块的存储数量及存储位置决策问题，以在满足数据可靠性的约束下获得平衡的数据存储和读取性能.

3 系统模型与问题定义

本节首先介绍基于纠删码的BFT联盟链，然后给出编码数据块的存储数量及存储位置决策问题的形式化定义.

3.1 基于纠删码的BFT联盟链

本文考虑一种基于纠删码的BFT联盟链，其容错率为1/3，节点数量为N，且联盟链中的节点根据其公钥大小从1到N编号，系统采用RS码（Reed-Solomon Code）对原始数据块进行编码. RS码是一种常见的纠删码，利用RS码可以将k个原始数据块编码成$k+m$个编码数据块，m是额外生成的校验块数量；在丢失不超过m个编码数据块的情况下，RS码可以解码恢复原始的k个数据块. 接下来简要说明基于纠删码的BFT联盟链的工作原理.

首先对联盟链的区块进行分组，每组包含$N'=N-\lfloor(N-1)/3\rfloor$个区块，然后对每组$N'$个区块使用RS码生成$N-N'$个校验块；对于不足$N'$个区块的分组，因为无法使用RS码对其进行编码处理，所以该分组中的区块在系统的每个节点中都会有一个存储备份. 为了方便描述，将第e组的N'个原始数据块及$N-N'$个校验块记为编码数据块集合$\mathcal{K}_e$. 对于$\mathcal{K}_e$中的每个编码数据块k，只选择部分区块链节点来存储k，从而降低了区块链的存储冗余. 例1是一个基于纠删码的BFT联盟链示例.

例1. 如图1所示，BFT联盟链由4个节点$m_1,\cdots\cdots,m_4$组成，容错率为1/3. 此时，$N=4$，$N'=3$，令$k=3$，$m=1$. 假设区块链网络中有6个区块$b_1,\cdots\cdots,b_6$，先将区块分成两组(b_1,b_2,b_3)和(b_4,b_5,b_6)，然后对每组区块进行编码. 例如第一组，使用b_1、b_2、b_3生成1个校验块$\tilde{b}_1^1$，然后分别存储在节点$m_1,\cdots\cdots,m_4$中. 在网络中存在四种数据块，分别为完整区块、仅有区块头的数据块、完整校验块以及仅含有校验块哈希值的数据块. 所有节点保存所有区块的区块头以及校验块的哈希值，以进行数据的验证，而完整区块和完整校验块仅存储在部分节点中，以减少存储冗余. 从图1可以看到，编码后每个节点仅需存储2个编码数据块，总计8个编码数据块；而原BFT联盟链需要存储24个区块，采用纠删

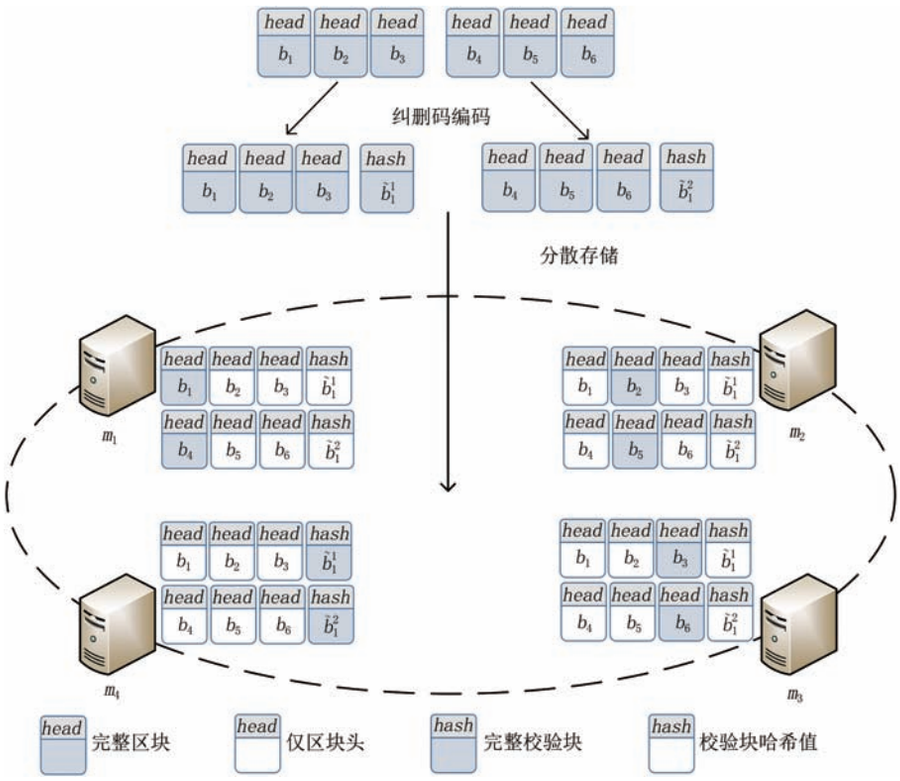

图1　基于纠删码的BFT联盟链示例

码编码存储后有效地减少了系统所需的存储空间.

但是,存储冗余的减少带来了数据安全性降低和数据读取延迟增大的问题,而编码数据块的存储数量及存储位置是影响数据的可靠性、存储代价和数据读取延迟的关键因素.因此,本文研究编码数据块的存储数量及存储位置决策问题,在满足数据可靠性的约束下获得平衡的数据存储和读取性能.本文所用符号见表1.

表1　本文符号表

符号	定义
$\mathcal{K}_e$	第e组编码数据块集合,包括原始数据块和校验块
$\mathcal{N}$	区块链的节点集合
s	单个编码数据块的存储代价
$l_{i,j}$	节点i与节点j之间的通信延迟
$p_{k,j}$	节点j对编码数据块k的访问概率
$y_{k,i}$	节点i是否存储编码数据块k(1表示存储,0表示未存储)
$x_{k,i,j}$	节点j是否从节点i读取编码数据块k(1表示是,0表示否)

3.1.1　安全问题

在对联盟链的编码数据块进行存储分配时,需要保证数据的可靠性.如果采用图1的编码数据块存储分配方案,BFT联盟链的数据是安全的.假设节点m_1为拜占庭节点,丢失的区块b_1、b_4分别可以通过b_2、b_3、校验块$\tilde{b}_1^1$和b_5、b_6、校验块$\tilde{b}_1^2$进行RS码解码恢复;假设节点m_2为拜占庭节点时,丢失的区块b_2、b_5分别可以通过b_1、b_3、校验块$\tilde{b}_1^1$和b_4、b_6、校验块$\tilde{b}_1^2$进行RS码解码恢复;当其他节点为拜占庭节点时,同样可以恢复丢失的数据.因此,此时BFT联盟链的数据是安全的.若改变存储分配方案,如节点m_2不存储区块b_2,节点m_1存储区块b_1和b_2;此时,若节点m_1为拜占庭节点,即第一组数据丢失了2个编码数据块,则超过了RS码的容错能力$m=1$,丢失的区块b_1、b_2将无法通过RS码进行恢复.为了保证BFT联盟链数据的可靠性,在确定编码数据块的存储位置时,必须满足以下约束:联盟链中任意$\lfloor(N-1)/3\rfloor$个节点为拜占庭节点时,每个编码数据块集合$\mathcal{K}_e$最多丢失$\left\lfloor\left(\left|\mathcal{K}_e\right|-1\right)/3\right\rfloor$个数据.

3.1.2　数据读取延迟

在基于纠删码的BFT联盟链中,由于每个节点都不再存储所有的区块副本,节点可能需要向其它节点请求需要的区块,获取数据的过程中可能需要经过多跳网络,从而导致了不可忽视的延迟.例如在图1中,节点m_3需要从节点m_1处获取区块b_1,b_4.当某些节点为拜占庭节点时,节点甚至需要获取全网大多数节点的数据,然后通过RS码进行数据恢复.如图1,假设节点m_1为拜占庭节点,节点m_3若想要获取区块b_1,则需要向节点m_2,m_4请求区块b_2和

校验块$\tilde{b}_1^1$进行解码恢复，从而导致较高的延迟.

编码数据块的存储数量越多，数据读取延迟就越低，但是编码数据块的存储代价却会升高.因此编码数据块存储数量与数据读取延迟之间存在折中关系，同时编码数据块的存储位置也会影响数据读取延迟.

3.2 问题定义

使用纠删码技术虽然可以有效地减少整个区块链系统所需的存储空间，但是数据存储冗余的减少也带来以下两个问题：1)数据读取性能降低；2)需要保证数据的可靠性.在对区块进行编码和存储时，必须要考虑到上述两个问题.特别地，在存储每组编码数据块$\mathcal{K}_e$时，本文通过决定编码数据块的存储位置、存储数量来实现数据存储代价和数据读取性能间的平衡，同时保证数据的可靠性.编码数据块的存储数量及存储位置决策问题的模型为

$$\min \sum_{k\in\mathcal{K}_e}\left(\lambda_1\sum_{i\in\mathcal{N}} sy_{k,i}+\lambda_2\sum_{i,j\in\mathcal{N}} L_{k,i,j}x_{k,i,j}\right) \tag{1}$$

s.t.

$$L_{k,i,j}=l_{i,j}\times p_{k,j}\quad i,j\in\mathcal{N},k\in\mathcal{K}_e \tag{2}$$

$$\sum_{i\in\mathcal{N}} x_{k,i,j}=1\quad j\in\mathcal{N},k\in\mathcal{K}_e \tag{3}$$

$$x_{k,i,j}\leqslant y_{k,i}\quad i,j\in\mathcal{N},k\in\mathcal{K}_e \tag{4}$$

$$|S'|\leqslant\left\lfloor\frac{|\mathcal{K}_e|-1}{3}\right\rfloor,\forall S'=\left\{S_1,S_2,\cdots,S_{\left\lfloor\frac{N-1}{3}\right\rfloor}\right\},$$

$$S_i=\left\{k\middle|\, y_{k,i}=1,k\in\mathcal{K}_e\right\} \tag{5}$$

$$\lambda_1,\lambda_2>0 \tag{6}$$

$$x_{k,i,j},y_{k,i}\in\{0,1\}\quad i,j\in\mathcal{N},k\in\mathcal{K}_e \tag{7}$$

目标公式(1)表示最小化所有编码数据块的存储和延迟的总代价，其中λ_1和λ_2分别表示存储代价和延迟代价的权重；约束(2)表示节点j从节点i读取编码数据块k的延迟代价；约束(3)表示节点j必须也只能与一个存储编码数据块k的节点存在映射关系，即节点j从该节点读取编码数据块k；约束(4)表示若节点j从节点i读取编码数据块k，则节点i必须存储编码数据块k；约束(5)表示联盟链中任意$\lfloor(N-1)/3\rfloor$个节点为拜占庭节点时，每个编码数据块集合K_e最多丢失$\lfloor(|\mathcal{K}_e|-1)/3\rfloor$个数据块，其中$S'$表示任意$\lfloor(N-1)/3\rfloor$个节点存储的编码数据块集合.

4 延迟感知的编码数据块分配算法

无容量限制的设施选址问题(Uncapacitated Facility Location Problem，UFLP)是经典的*NP*-hard问题之一.UFLP问题的定义如下：$\mathcal{F}$表示设施集合，$\mathcal{C}$表示顾客集合，对于任意的$i'\in\mathcal{F},j'\in\mathcal{C}$，$f_{i'}$表示设施$i'$的开设费用，$\mathcal{C}_{i',j'}$表示设施$i'$为顾客$j'$提供服务的费用，问题的目标是选择$\mathcal{F}$的子集$\mathcal{F}'$，开设$\mathcal{F}'$中的设施，同时将$\mathcal{C}$中的顾客分派到开设的设施中，使得总费用(开设费用和服务费用)最小[34].本文的问题模型中，区块链的节点集合$\mathcal{N}$和编码数据块集合$\mathcal{K}_e$分别对应UFLP问题中的$\mathcal{F}$和$\mathcal{C}$，节点保存编码数据块的存储成本对应设施的开设费用，节点对编码数据块的访问延迟成本对应设施为顾客提供服务的费用，本文的编码数据块存储数量和存储位置决策问题的目标是选择节点子集存储编码数据块，同时将用户对编码数据块的访问请求分派到合适的存储该数据块的节点上，使得总成本(存储成本和延迟成本)最小.由上可见，本文的问题可归约为UFLP问题，从而说明本文的问题也是一个*NP*-hard问题.

本节针对编码数据块的存储数量及存储位置决策问题，提出了延迟感知的编码数据块分配算法(Latency-aware Encoded data chunks Allocation algorithm，LEA).算法LEA决定编码数据块(原始数据块与校验块)的存储数量及存储位置，从而在实现数据存储代价和读取性能平衡的同时保证联盟链数据的可靠性.

图2描述了算法LEA的基本流程.算法首先求解编码数据块的存储数量及存储位置决策问题的松弛问题以及该松弛问题的对偶问题，然后依次为每个编码数据块确定其存储数量和存储位置，从而得到初始的编码数据块存储分配方案：1)根据对偶问题的最优解选取某个节点作为聚类中心，然后根据松弛问题的最优解将某些节点与该节点放入同一聚类中，重复此过程直到所有节点均属于某个聚类；2)根据松弛问题的最优解，在每个聚类中随机选择至少一个节点存储该编码数据块的备份.最后调整得到的初始编码数据块存储分配方案使其满足被松弛的约束条件，并将每个节点分配到最近的存储有该编码数据块的节点进行数据读取.

4.1 编码数据块初始分配

首先，对编码数据块的存储数量及存储位置决

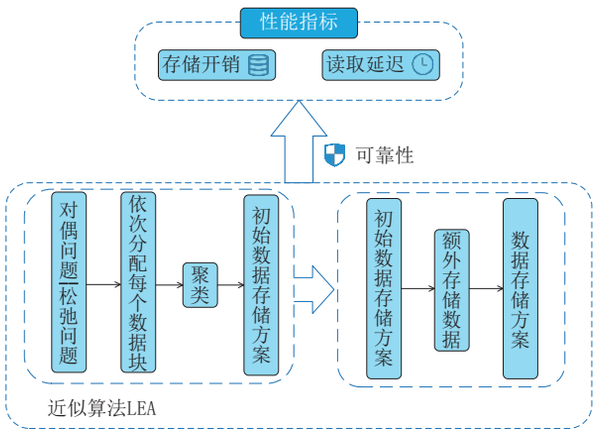

图2　编码数据块分配

策问题进行松弛处理.编码数据块的存储数量及存储位置决策问题的优化模型见公式(1),为了使问题易于解决,先不考虑可靠性约束(5),从而得到如下的整数规划问题,然后借鉴设施选址问题[34],利用松弛技术与对偶技术进行求解.

$$\min \sum_{k\in\mathcal{K}_e}\left(\lambda_1\sum_{i\in\mathcal{N}} s y_{k,i}+\lambda_2\sum_{i,j\in\mathcal{N}} L_{k,i,j}x_{k,i,j}\right) \tag{8}$$

s.t.

$$L_{k,i,j}=l_{i,j}\times p_{k,j}\quad i,j\in\mathcal{N},k\in\mathcal{K}_e \tag{9}$$

$$\sum_{i\in\mathcal{N}} x_{k,i,j}=1\quad j\in\mathcal{N},k\in\mathcal{K}_e \tag{10}$$

$$x_{k,i,j}\leqslant y_{k,i}\quad i,j\in\mathcal{N},k\in\mathcal{K}_e \tag{11}$$

$$\lambda_1,\lambda_2>0 \tag{12}$$

$$x_{k,i,j},y_{k,i}\in\{0,1\}\quad i,j\in\mathcal{N},k\in\mathcal{K}_e \tag{13}$$

对整型变量进行松弛得到如下线性规划问题:

$$\min \sum_{k\in\mathcal{K}_e}\left(\lambda_1\sum_{i\in\mathcal{N}} s y_{k,i}+\lambda_2\sum_{i,j\in\mathcal{N}} L_{k,i,j}x_{k,i,j}\right) \tag{14}$$

s.t.

$$L_{k,i,j}=l_{i,j}\times p_{k,j}\quad i,j\in\mathcal{N},k\in\mathcal{K}_e \tag{15}$$

$$\sum_{i\in\mathcal{N}} x_{k,i,j}=1\quad j\in\mathcal{N},k\in\mathcal{K}_e \tag{16}$$

$$x_{k,i,j}\leqslant y_{k,i}\quad i,j\in\mathcal{N},k\in\mathcal{K}_e \tag{17}$$

$$\lambda_1,\lambda_2>0 \tag{18}$$

$$x_{k,i,j},y_{k,i}\geqslant 0\quad i,j\in\mathcal{N},k\in\mathcal{K}_e \tag{19}$$

该线性规划问题的对偶问题为:

$$\max \sum_{k\in\mathcal{K}_e}\sum_{j\in\mathcal{N}}\alpha_{k,j} \tag{20}$$

s.t.

$$\alpha_{k,j}-\beta_{k,i,j}\leqslant\lambda_2 L_{k,i,j}\quad i,j\in\mathcal{N},k\in\mathcal{K}_e \tag{21}$$

$$\sum_{j\in\mathcal{N}}\beta_{k,i,j}\leqslant\lambda_1 s\quad i\in\mathcal{N},k\in\mathcal{K}_e \tag{22}$$

$$L_{k,i,j}=l_{i,j}\times p_{k,j}\quad i,j\in\mathcal{N},k\in\mathcal{K}_e \tag{23}$$

$$\lambda_1,\lambda_2>0 \tag{24}$$

$$\alpha_{k,j},\beta_{k,i,j}\geqslant 0\quad i,j\in\mathcal{N},k\in\mathcal{K}_e \tag{25}$$

定义1.　设$(\tilde{x},\tilde{y})$是线性规划(14)的可行解,对任意$k\in\mathcal{K}_e,i,j\in\mathcal{N}$,若$\tilde{x}_{k,i,j}>0$,且$\tilde{x}_{k,i,j}=\tilde{y}_{k,i}$,则称$(\tilde{x},\tilde{y})$是线性规划(14)的完全解.

算法1.　构造完全解.

输入:可行解$(\bar{x},\bar{y})$

输出:完全解$(\tilde{x},\tilde{y})$

1. 复制$(\bar{x},\bar{y})$得到$(\tilde{x},\tilde{y})$;
2. WHILE存在区块k及节点i的组合(k,i),使$0<\tilde{x}_{k,i,j}<\tilde{y}_{k,i},\exists j\in\mathcal{N}$ DO
3. 令$j_1\leftarrow\arg\min_{j\in\mathcal{N};0<\tilde{x}_{k,i,j}<\tilde{y}_{k,i}}\tilde{x}_{k,i,j}$;
4. 设置虚拟节点i'指向节点i,即节点i'和i代表同一节点且存储代价和位置相同,令$\tilde{y}_{k,i'}\leftarrow\tilde{y}_{k,i}-\tilde{x}_{k,i,j_1},\tilde{y}_{k,i}\leftarrow\tilde{x}_{k,i,j_1}$;
5. FOR每一个节点$j\in\mathcal{N}$ DO
6. IF $\tilde{x}_{k,i,j}>0$ THEN
7. 令$\tilde{x}_{k,i',j}\leftarrow\tilde{x}_{k,i,j}-\tilde{x}_{k,i,j_1},\tilde{x}_{k,i,j}\leftarrow\tilde{x}_{k,i,j_1}\leftarrow\tilde{y}_{k,i}$;
8. END IF
9. END FOR
10. END WHILE
11. RETURN$(\tilde{x},\tilde{y})$

算法1可以利用线性规划(14)的可行解$(\bar{x},\bar{y})$构造出对应的完全解$(\tilde{x},\tilde{y})$,并且该完全解的分数存储代价和分数延迟代价与$(\bar{x},\bar{y})$的对应代价分别相等.若不存在区块k及节点i的组合(k,i),使$0<\tilde{x}_{k,i,j}<\tilde{y}_{k,i},\exists j\in\mathcal{N}$,则可行解即为完全解;否则经过3-9步,$(\tilde{x},\tilde{y})$中满足$0<\tilde{x}_{k,i,j}<\tilde{y}_{k,i},\exists j\in\mathcal{N}$的组合$(k,i,j)$至少减少了一对.因为这样的组合最多有$|\mathcal{K}||\mathcal{N}|^2$种,所以在多项式时间内可以构造出对应的完全解.算法1的3-9步将节点i逻辑上拆分为i和i',且存储代价和位置相同,显然算法所得的完全解$(\tilde{x},\tilde{y})$是线性规划(14)的可行解,且分数存储代价和分数延迟代价与$(\bar{x},\bar{y})$的对应代价分别相等.

接下来给出编码数据块的初始存储分配算法,如算法2所示.

算法2.　编码数据块的初始存储分配.

输入:待存储分配的编码数据块集合$\mathcal{K}_e$,区块链节点集合$\mathcal{N}$,编码数据块存储代价s,区块链节点间延迟,区块链节点对编码数据块的访问概率

输出:编码数据块的存储分配方案(x,y)

1. 求解线性规划(14)及其对偶规划(20)的最优解,并利用算法1获得与线性规划(14)最优解对应的完全解,记为(x^*,y^*)和(α^*,β^*);
2. 初始化编码数据块的存储分配方案(x,y),令所

有分量为0；

3. FOR每一个待存储分配的编码数据块$k\in\mathcal{K}_e$ DO

4. 定义聚类中心集合$\mathcal{C}\leftarrow\varnothing$，节点集合$\mathcal{N}'\leftarrow\mathcal{N}$，对所有节点$j$，定义$\mathcal{F}_{k,j}\leftarrow\left\{i\middle|x^*_{k,i,j}>0,i\in\mathcal{N}\right\}$；

5. WHILE $\mathcal{N}'\neq\varnothing$ DO

6. 选择$\mathcal{N}'$中$\alpha^*_{k,j}$最小的节点作为聚类中心j'，令$\mathcal{C}\leftarrow\mathcal{C}\cup\{j'\}$；

7. 定义$B_{k,j'}\leftarrow\left\{j\middle|\mathcal{F}_{k,j}\cap\mathcal{F}_{k,j'}\neq\varnothing,j\in\mathcal{N}'\right\}$为以$j'$为中心的聚类，令$\mathcal{N}'\leftarrow\mathcal{N}'\backslash B_{k,j'}$；

8. END WHILE

9. 对所有节点$j'\in\mathcal{C}$，以概率$\ln 3\times x^*_{k,i,j'}$选择$\mathcal{F}_{k,j'}$中的一个节点i存储编码数据块k，即令$y_{k,i}\leftarrow 1$；

10. 记$\mathcal{R}\leftarrow\mathcal{N}\backslash\bigcup_{j'\in\mathcal{C}}\mathcal{F}_{k,j'}$，对任意节点$i\in\mathcal{R}$，以概率$\ln 3\times y^*_{k,i}$独立的选择节点$i$存储编码数据块$k$，即令$y_{k,i}\leftarrow 1$；

11 令每个节点j从最近的存储编码数据块k的节点i读取数据；

12. END FOR

13. RETURN(x,y)

算法2首先求解线性规划(14)及其对偶规划(20)的最优解(如单纯形法)，并求得线性规划(14)的最优解对应的完全解.然后依次为每个编码数据块k确定其存储数量和存储位置：1)选择分摊代价最小($\alpha^*_{k,j}$值最小)的节点作为聚类中心，然后将与其分数连接存在重合的节点与该节点放入同一聚类中，重复此过程直到所有节点均属于某个聚类；2)根据线性规划(14)的最优解，以概率$\ln 3\times x^*_{k,i,j'}$选择一个与聚类中心j'分数连接的节点i存储区块k；3)对于未与任何聚类中心分数相连的节点i，以概率$\ln 3\times y^*_{k,i}$独立地选择节点i存储区块k；重复此过程，直到所有的编码数据块分配完毕.

4.2 延迟感知的编码数据块分配算法

由算法2得到初始的编码数据块存储分配方案，但该初始方案并不能保证满足编码数据块存储数量及存储位置决策问题的约束条件(5)，接下来对初始编码数据块存储分配方案进行调整并给出延迟感知的编码数据块分配算法LEA，如算法3所示.

算法3. 算法LEA.

输入：待存储分配编码数据块集合$\mathcal{K}_e$，区块链节点集合$\mathcal{N}$，编码数据块存储代价s，区块链节点间延迟，区块链节点对编码数据块的访问概率

输出：编码数据块的存储分配方案(x,y)

1. 通过算法2初始化编码数据块的存储分配方案(x,y)；

2. 为每个节点定义集合$S_i\leftarrow\{k\,|\,y_{k,i}=1,\ k\in\mathcal{K}_e\}$，定义待分配的编码数据块集合$\mathcal{K}'\leftarrow\mathcal{K}_e$，节点集合$\mathcal{N}'\leftarrow\mathcal{N}$；

3. 为每个节点定义集合$O_i\leftarrow S_i\backslash\bigcup_{j\in\mathcal{N}',j\neq i}(S_j\cap S_i)$；

4. 将所有节点$i\in\mathcal{N}'$按照$|O_i|$的大小非递减排序，形成序列Q；

5. FOR 序列Q中的每一个节点i DO

6. IF $|O_i\cap\mathcal{K}'|>0$ DO

7. 等概率随机选择$O_i\cap\mathcal{K}'$中的一个元素k'，令$\mathcal{K}'\leftarrow\mathcal{K}'\backslash\{k'\}$；

8. ELSE IF

9. 从$\mathcal{K}'$中等概率随机选择编码数据块k'，在节点i上存储编码数据块k'，令$y_{k',i}\leftarrow 1$，$\mathcal{K}'\leftarrow\mathcal{K}'\backslash\{k'\}$；

10. END IF

11. END FOR

12. FOR每一个编码数据块$k\in\mathcal{K}_e$ DO

13. 将每个节点j重新连接到最近的存储编码数据块k的节点i；

14. END FOR

15. RETURN(x,y)

算法LEA通过额外存储编码数据块的方式，使存储分配方案满足约束条件(5).在算法LEA的步骤5-11中，每次迭代都会选择一个节点i存储一个编码数据块k'，然后将节点i和编码数据块k'从各自的集合中去除.因此在循环结束后，联盟链中的所有节点存储了互不相同的区块.不妨将节点i存储一个编码数据块k'表示为组合(i,k')，则存在N个这样的组合且所有组合中任意对应元素均不相同.在网络中任意$\lfloor(N-1)/3\rfloor$个节点失效后，网络中仍然有$N-\lfloor(N-1)/3\rfloor$个节点有效，即$N-\lfloor(N-1)/3\rfloor$个组合(i,k')仍然存在，此时网络中至少仍存在$N-\lfloor(N-1)/3\rfloor$块不同的编码数据块.考虑到编码方式$N=|\mathcal{K}_e|$，因此网络中至少仍存在$|\mathcal{K}_e|-\lfloor(|\mathcal{K}_e|-1)/3\rfloor$块不同的编码数据块，即最多丢失了$\lfloor(|\mathcal{K}_e|-1)/3\rfloor$个编码数据块，约束条件(5)得到满足.

4.3 算法性能分析

引理1. 假设(x^*,y^*)是线性规划(14)的最优解，对于任意节点$i,j\in\mathcal{N}$，如果$x_{k,i,j}>0$，则有$\lambda_2 L_{k,i,j}\leqslant\alpha^*_{k,j},\forall k\in\mathcal{K}_e$.

证明. 对偶问题满足如下的互补松弛条件：

$$x_{k,i,j}\left(\lambda_2 L_{k,i,j}-\alpha^*_{k,j}+\beta^*_{k,i,j}\right)=0.$$

因为$x_{k,i,j}>0$，则$\lambda_2 L_{k,i,j}-\alpha^*_{k,j}=-\beta^*_{k,i,j}$，又因为

$\beta^*_{k,i,j}\geqslant 0$，也即$\lambda_2 L_{k,i,j}-\alpha^*_{k,j}\leqslant 0$，所以证得$\lambda_2 L_{k,i,j}\leqslant \alpha^*_{k,j}, \forall k\in\mathcal{K}_e$. 证毕.

定义2. 对于线性规划(14)的最优解(x^*,y^*)，将编码数据块k的存储代价S^*_k、读取编码数据块k的延迟代价$\mathcal{L}^*_k$、最优解的累积代价$\mathcal{LP}^*$分别定义如下：

$$S^*_k=\sum\nolimits_{i\in\mathcal{N}} s\times y^*_{k,i}$$

$$\mathcal{L}^*_k=\sum\nolimits_{i,j\in\mathcal{N}} L_{k,i,j}\times x^*_{k,i,j}$$

$$\mathcal{LP}^*=\sum\nolimits_{k\in\mathcal{K}_e}\Big(\lambda_1\sum\nolimits_{i\in\mathcal{N}} s\times y^*_{k,i}+\lambda_2\sum\nolimits_{i,j\in\mathcal{N}} L_{k,i,j}\times x^*_{k,i,j}\Big)=\sum\nolimits_{k\in\mathcal{K}_e}\big(\lambda_1\mathcal{S}^*_k+\lambda_2\mathcal{L}^*_k\big)$$

引理2. 算法2所得存储分配方案中单个编码数据块k的存储代价期望如下：

$$E\big[\mathcal{S}_k\big]=\sum\nolimits_{i\in\mathcal{N}} s\times\ln 3\times y^*_{k,i}=\ln 3\times\mathcal{S}^*_k.$$

证明. 分以下两种情况进行讨论：

(1) 对于节点$i\in\bigcup_{j'\in\mathcal{C}}\mathcal{F}_{k,j'}$，由算法2步骤9可以知道节点$i$存储编码数据块$k$的概率为$\ln 3\times x^*_{k,i,j'}$，其中$j'\in\mathcal{C}$，因为$(x^*,y^*)$是完全解，有$x^*_{k,i,j'}=y^*_{k,i}$.

(2) 对于节点$i\in\mathcal{N}\backslash\bigcup_{j'\in\mathcal{C}}\mathcal{F}_{k,j'}$，由算法2步骤10可以知道节点$i$存储编码数据块$k$的概率为$\ln 3\times y^*_{k,i}$.

因此，任意节点i存储编码数据块k的概率为$\ln 3\times y^*_{k,i}$，所以算法2所得存储分配方案中编码数据块k的存储代价期望为：$E\big[\mathcal{S}_k\big]=\sum_{i\in\mathcal{N}} s\times\ln 3\times y^*_{k,i}=\ln 3\times \mathrm{S}^*_k$. 证毕.

引理3. 算法2所得存储分配方案中，所有节点对编码数据块k的读取延迟代价期望满足：

$$E\big[\mathcal{L}_k\big]\leqslant\ln 3\times\mathcal{L}^*_k+\sum\nolimits_{i\in\mathcal{N}}\alpha^*_{k,j}/\lambda_2.$$

证明. 首先分析单个节点读取编码数据块k的读取延迟代价期望，分为以下两种情况：

(1) 对于节点$j\in\mathcal{C}$，由算法2步骤9可知与节点j分数相连的节点i存储编码数据块k的概率为$\ln 3\times x^*_{k,i,j}$，因此节点j读取编码数据块k的代价期望为$E\big[\mathcal{L}_{k,j}\big]=\sum_{i\in\mathcal{F}_{k,j}}\ln 3\times x^*_{k,i,j}\times L_{k,i,j}$，又因对于$i\notin\mathcal{F}_{k,j}$，$x^*_{k,i,j}=0$，所以有$E\big[\mathcal{L}_{k,j}\big]=\sum_{i\in\mathcal{F}_{k,j}}\ln 3\times x^*_{k,i,j}\times L_{k,i,j}=\ln 3\times\sum_{i\in\mathcal{N}} x^*_{k,i,j}\times L_{k,i,j}$.

(2) 对于节点$j\notin\mathcal{C}$，节点j存在某个聚类中，故存在$j'\in\mathcal{C}$，使得$\mathcal{F}_{k,j}\cap\mathcal{F}_{k,j'}\neq\varnothing$，记$\mathrm{C}'=\{j'\in\mathcal{C}|\mathcal{F}_{k,j}\cap\mathcal{F}_{k,j'}\neq\varnothing\}$. 可以知道一定存在节点$j'\in\mathcal{C}$，使得$\big|\mathcal{F}_{k,j}\cap\mathcal{F}_{k,j'}\big|\geqslant 1$. 考虑如下一般情况，见图3.

对聚类中心节点$j'\in\mathcal{C}'$，令事件$E_{k,j'}$表示$\mathcal{F}_{k,j}\cap\mathcal{F}_{k,j'}$中有一节点存储编码数据块$k$，由算法2步骤9可以得知事件$E_{k,j'}$的发生概率为$\tilde{p}_{k,j'}=\sum_{i\in\mathcal{F}_{k,j}\cap\mathcal{F}_{k,j'}}\ln 3\times x^*_{k,i,j'}$，且事件$E_{k,j'}, j'\in\mathcal{C}'$相互独立；若事件$E_{k,j'}$已经发生，则以概率$\ln 3\times x^*_{k,i,j'}/\tilde{p}_{k,j'}$选择节点$i\in\mathcal{F}_{k,j}\cap\mathcal{F}_{k,j'}$存储编码数据块$k$，因此事件$E_{k,j'}$对应的读取延迟代价期望为$\tilde{L}_{k,j'}=\sum_{i\in\mathcal{F}_{k,j}\cap\mathcal{F}_{k,j'}}\ln 3\times L_{k,i,j'}\times x^*_{k,i,j'}/\tilde{p}_{k,j'}$. 令事件$E_{k,i}$表示节点$i\in\mathcal{F}_{k,j}\cap\mathcal{R}$存储编码数据块$k$，由算法2步骤10可知事件$E_{k,i}$的发生概率为$\tilde{p}_{k,i}=\ln 3\times y^*_{k,i}$，且事件$E_{k,i}, i\in\mathcal{F}_{k,j}\cap\mathcal{R}$相互独立，事件对应的读取延迟代价为$\tilde{L}_{k,i}=L_{k,i,j}$. 由算法可知，所有事件$E_{k,v}$相互独立，$\sum_v\tilde{p}_{k,v}=\sum_{j'\in\mathcal{C}'}\tilde{p}_{k,j'}+\sum_{i\in\mathcal{F}_{k,j}\cap\mathcal{R}}\tilde{p}_{k,i}$，由完全解定义可知$\sum_v\tilde{p}_{k,v}=\ln 3$并且$\sum_v\tilde{p}_{k,v}\times\tilde{L}_{k,v}=\ln 3\times\sum_{i\in\mathcal{N}} x^*_{k,i,j}\times L_{k,i,j}$.

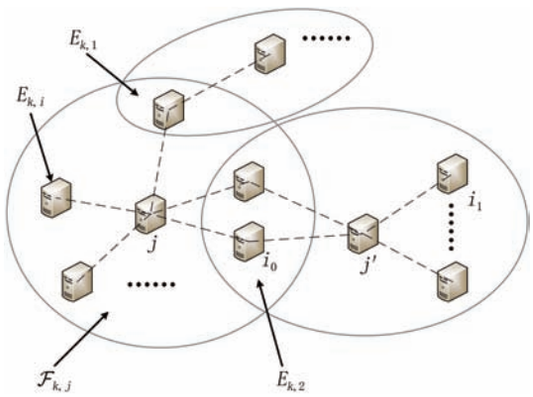

图3　节点$j\notin\mathcal{C}$情况下数据读取延迟期望分析

不失一般性，假设存在d个事件$E_{k,v}$，$v\in\{1,2,\cdots,d\}$且$\tilde{L}_{k,1}\leqslant\tilde{L}_{k,2}\leqslant\cdots\leqslant\tilde{L}_{k,d}$. 定义事件$\overline{E_k}$表示$d$个事件都没有发生，则其概率为$\tilde{q}=\prod_{v=1}^{d}\big(1-\tilde{p}_{k,v}\big)$，通过不等式$1+x\leqslant e^x, x>0$，可以得知$\tilde{q}=\prod_{v=1}^{d}\big(1-\tilde{p}_{k,v}\big)\leqslant\prod_{v=1}^{d}e^{-\tilde{p}_{k,v}}=e^{-\sum_{v=1}^{d}\tilde{p}_{k,v}}=e^{-\ln 3}=1/3$. 考虑节点$j$从最近的存储数据块$k$的节点读取数据. 若事件$E_{k,1}$发生，其概率为$\tilde{p}_{k,1}$，则节点$j$读取编码数据块$k$的延迟代价期望为$\tilde{L}_{k,1}$；若事件$E_{k,1}$未发生，事件$E_{k,2}$发生，其概率为$\big(1-\tilde{p}_{k,1}\big)\tilde{p}_{k,2}$，则节点$j$读取编码数据块$k$的延迟代价期望为$\tilde{L}_{k,2}$，以此类推. 事件$\overline{E_k}$发生，其概率为$\tilde{q}$，节点$j$读取编码数据块$k$的延迟代价期望不超过$3\alpha^*_{k,j}/\lambda_2$，原因如下：

事件$\overline{E_k}$的发生说明与节点j分数相连的节点集合$\mathcal{F}_{k,j}$均未存储编码数据块k. 对于聚类中心节点j'，由算法2步骤9可知$\mathcal{F}_{k,j'}$中必有一节点存储编码数据块k，不妨假设该节点为i_1，如图3所示. 节点j可以从节点i_1处读取编码数据块k，此时节点j读取编

码数据块k的延迟代价不超过$L_{k,i_0,j}+L_{k,i_0,j'}+L_{k,i_1,j'}$(本文假设延迟代价满足三角不等式),由引理1可知$\lambda_2\times L_{k,i,j}\leqslant\alpha^*_{k,j}$,同时由算法2步骤6可知$\alpha^*_{k,j'}\leqslant\alpha^*_{k,j}$,所以$L_{k,i_0,j}+L_{k,i_0,j'}+L_{k,i_1,j'}\leqslant 2\alpha^*_{k,j'}/\lambda_2+\alpha^*_{k,j}/\lambda_2\leqslant 3\alpha^*_{k,j}/\lambda_2$.

通过上述分析可知,节点j读取编码数据块k的延迟代价期望不可能超过下面的值,即:

$$\tilde{p}_{k,1}\tilde{L}_{k,1}+(1-\tilde{p}_{k,1})\tilde{p}_{k,2}\tilde{L}_{k,2}+\cdots+(1-\tilde{p}_{k,1})(1-\tilde{p}_{k,2})\cdots\tilde{p}_{k,d}\tilde{L}_{k,d}+3\tilde{q}\alpha^*_{k,j}/\lambda_2\leqslant\tilde{p}_{k,1}\tilde{L}_{k,1}+\tilde{p}_{k,2}\tilde{L}_{k,2}+\cdots+\tilde{p}_{k,d}\tilde{L}_{k,d}+3\tilde{q}\alpha^*_{k,j}/\lambda_2\leqslant\ln 3\times\sum_{i\in\mathcal{N}}x^*_{k,i,j}\times L_{k,i,j}+\alpha^*_{k,j}/\lambda_2.$$

综合以上两种情况可知,单个节点j读取编码数据块k的读取延迟代价期望不超过$\ln 3\times\sum_{i\in\mathcal{N}}x^*_{k,i,j}\times L_{k,i,j}+\alpha^*_{k,j}/\lambda_2$.因此,所有节点读取编码数据块$k$的读取延迟代价期望不超过$\sum_{j\in\mathcal{N}}\left(\ln 3\times\sum_{i\in\mathcal{N}}x^*_{k,i,j}\times L_{k,i,j}+\alpha^*_{k,j}/\lambda_2\right)=\ln 3\times\mathcal{L}^*_k+\sum_{j\in\mathcal{N}}\alpha^*_{k,j}/\lambda_2$. 证毕.

引理4. 算法2是公式(8)-(13)定义的编码数据块存储数量及存储位置决策整数规划问题的$\ln 3+1$近似算法.

证明. 假设$\mathcal{IP}^*$是整数规划(8)最优解的总代价,因线性规划(14)是整数规划(8)的松弛问题(目标函数相同,整数规划(8)的约束条件更加严格),有$\mathcal{LP}^*\leqslant\mathcal{IP}^*$.根据引理2和引理3可知,算法2所得解的累积代价期望满足:

$$\sum_{k\in\mathcal{K}_e}\left(\lambda_1E[\mathcal{S}_k]+\lambda_2E[\mathcal{L}_k]\right)\leqslant\ln 3\times\sum_{k\in\mathcal{K}_e}\left(\lambda_1\mathcal{S}^*_k+\lambda_2\mathcal{L}^*_k\right)+\sum_{k\in\mathcal{K}_e}\sum_{j\in\mathcal{N}}\alpha^*_{k,j}$$

由强对偶性可知$\sum_{k\in\mathcal{K}_e}\sum_{j\in\mathcal{N}}\alpha^*_{k,j}=\mathcal{LP}^*$,故$\sum_{k\in\mathcal{K}_e}\left(\lambda_1E[\mathcal{S}_k]+\lambda_2E[\mathcal{L}_k]\right)\leqslant(\ln 3+1)\mathcal{LP}^*\leqslant(\ln 3+1)\times\mathcal{IP}^*$.

因此算法2是关于整数规划(8)的$\ln 3+1$近似算法. 证毕.

定理1. 算法3是关于编码数据块的存储数量及存储位置决策问题的$\ln 3+2$近似算法.

证明. 本文已经说明了算法3的存储分配方案可以满足约束条件(5),接下来讨论算法3的调整对解的累积代价的影响.

算法3采用额外存储编码数据块的方法使存储分配方案满足约束条件(5),显然额外存储编码数据块不会导致读取延迟代价增大,因此只需考虑编码数据块存储代价的增加.算法3对存储分配方案的调整发生在步骤7和步骤9,因为步骤7并没有额外存储编码数据块,所以算法3所得的存储分配方案最坏增加$|\mathcal{K}_e|\times s$的存储代价.整数规划(8)的最优解必须保证所有$|\mathcal{K}_e|$个编码数据块都至少存储一份,否则目标函数中延迟代价将无限大且无法满足约束条件(10)和(11),所以整数规划(8)的最优解的存储代价一定大于等于$|\mathcal{K}_e|\times s$,且$\lambda_1|\mathcal{K}_e|\times s\leqslant\mathcal{IP}^*$.

假设算法3所得解的目标函数值为$\mathcal{P}^*$,原问题(1)的最优解的目标函数值为$\mathcal{OPT}^*$.因为整数规划(8)是原问题(1)的松弛问题,所以有$\mathcal{IP}^*\leqslant\mathcal{OPT}^*$,算法3所得解的累积代价满足:

$$\mathcal{P}^*\leqslant(\ln 3+1)\mathcal{IP}^*+\lambda_1|\mathcal{K}_e|\times s\leqslant(\ln 3+2)\mathcal{IP}^*\leqslant(\ln 3+2)\mathcal{OPT}^*.$$

因此,算法3是编码数据块的存储数量及存储位置决策问题的$\ln 3+2$近似算法. 证毕.

4.4 算法时间复杂度分析

本文假设联盟链中的节点数量为N,第e组的编码数据块集合为$\mathcal{K}_e$,根据RS码的编码方案可知,编码数据块集合$\mathcal{K}_e$中的原始区块数量为$N-\lfloor(N-1)/3\rfloor$,校验块的数量为$\lfloor(N-1)/3\rfloor$,因而$|\mathcal{K}_e|=N$.算法1对每一种区块k和节点i的组合(k,i)最多只需执行一次循环操作,将N个可行解全部转化为完全解,该循环操作需要时间$O(N)$.因为$k\in\mathcal{K}_e,i\in\mathcal{N}$,故算法1的时间复杂度为$O(|\mathcal{K}_e|N^2)=O(N^3)$.算法2先调用算法1以获得线性规划(14)的完全解,再针对每一个编码数据块k执行步骤4-11来选择数据块k的存储位置.因为对元素个数为N的两个集合进行操作(求并集、交集和差集)最快需要时间$O(N\log N)$,所以算法2的时间复杂度为$O(N^3+|\mathcal{K}_e|N^3\log N)=O(N^4\log N)$.算法3也即算法LEA先调用算法2来初始化编码数据块的存储分配方案,然后确定额外存储编码数据块的位置以满足约束条件(5),从而获得编码数据块的存储数量和存储位置决策问题的近似解.确定额外存储编码数据块的位置需要时间$O(N^3\log N)$,因而算法LEA的时间复杂度为$O(N^4\log N+N^3\log N)=O(N^4\log N)$.

5 实验数据和结果

为了对算法LEA的性能进行全面的评估,本文

分别在仿真环境和真实的联盟链系统中进行了实验验证，其中，真实环境下的实验是在Tendermint①平台上完成的，该平台是一个采用PBFT共识协议的开源联盟链系统.

5.1　实验设置

在实验中，区块大小设置为1 MB[14]，每个节点拥有100 MB的存储容量.在网络层，每个节点都与其对等节点保持TCP连接，以便所有节点都可以通过P2P协议相互通信.

为了评估算法LEA的性能，本文将文献[14]所提出的编码数据块存储分配方案BFTS和区块存储策略f-replica[31]作为对比算法.算法BFTS固定每个编码数据块的存储数量，然后生成一组与编码数据块的存储数量相同的随机数，根据随机数以及节点编号确定编码数据块的存储位置.算法f-replica将每个区块存储在至少$f+1$个节点上，其中f是系统中恶意节点的最大数量，该算法可以确保系统中至少有一个诚实节点会保存完整的区块，不会发生数据的丢失.

实验中使用平均单区块存储代价和平均延迟代价两个指标来衡量算法的性能[14]，其中平均单区块存储代价等于联盟链总存储代价除以原始数据块的数量(不包括生成的校验块)，平均延迟代价是所有节点对所有区块的读取延迟代价的平均值.因为各算法产生的平均延迟代价与节点对编码数据块的读取概率相关，为了更全面的评估算法的数据读取性能，在实验中采用均匀分布和重尾分布两种方式随机生成节点对编码数据块的读取概率[14].此外，为了平衡编码数据块存储代价和数据读取延迟，实验中设置算法LEA的存储代价和延迟代价权重$\lambda_1+\lambda_2=1$.同时，考虑到区块存储代价、可靠性和数据读取延迟性能间的折中，将算法BFTS中每个编码数据块的存储数量设置为2.实验参数详见表2.

表2　实验参数表

实验参数	取值范围
节点数量	5-40
恶意节点数量	0-10
区块大小	1 MB
单节点存储容量	100 MB

5.2　仿真环境中进行实验验证

在仿真实验中，为了模拟P2P网络，使用斯坦福网络分析平台(Stanford Network Analysis Platform，SNAP)②生成符合小世界模型的网络拓扑结构.包括互联网的底层架构、社交网络以及基因网络等在内的网络拓扑结构都显示出了小世界现象，因此本文在仿真实验中采用小世界模型生成网络拓扑.

5.2.1　节点数量对存储代价和延迟代价的影响

图4显示了各算法产生的平均单区块存储代价与不同节点数量的关系，算法LEA的权重$\lambda_1=0.8$、$\lambda_2=0.2$.从图4可以看出，算法LEA和算法BFTS产生的平均单区块存储代价随着节点数量的增加在一定范围内波动，而算法f-replica的平均单区块存储代价随着节点数量的增加而线性上升.这表明，基于纠删码的数据存储方案有效地减少了区块的存储冗余.算法LEA和算法BFTS在每轮编码中会对$N'=N-\lfloor(N-1)/3\rfloor$个区块进行处理，当区块数量为$N'$的整数倍时，所有区块都能使用RS纠删码进行编码存储，每个区块的存储代价得以有效降低；反之，最后一组的区块数量会达不到编码的要求，无法使用RS码对其进行编码，每个节点都会存储未编码区块的一个备份，从而导致这些区块的存储代价较高.因此，算法LEA和算法BFTS的平均单区块存储代价会随着节点数量的增加出现周期性波动.而算法f-replica将区块存储在至少$\lfloor(N-1)/3\rfloor+1$个节点上，导致该算法的平均单区块存储代价与节点数量成线性关系.

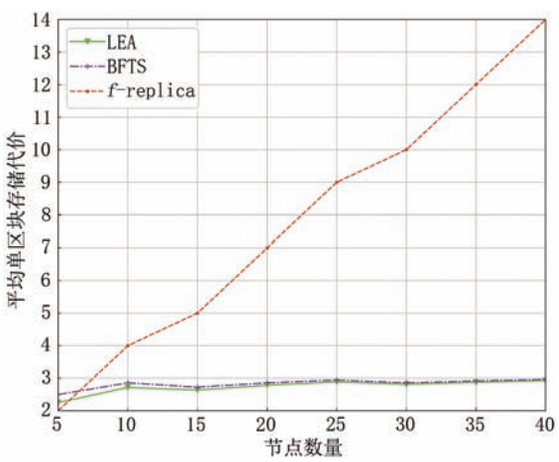

图4　节点数量对平均单区块存储代价的影响

从图4还可以看出，在节点数量相同的情况下，算法LEA相比算法BFTS产生的平均单区块存储代价更低，最好时优于算法BFTS 11.1%.这是因为算法LEA根据目标函数权衡存储代价和数据读取延迟，确定编码数据块的存储数量，从而优化了编码数据块的存储代价.此外，算法LEA的平均单区块

① Tendermint. https://github.com/tendermint/tendermint/
② Stanford Network Analysis Project. http://snap.stanford.edu

存储代价显著低于算法f-replica.例如,当节点数量为40时,算法LEA的平均单区块存储代价比算法f-replica低3.78倍.

图5显示了不同节点数量对区块链存储能力的影响,其中算法LEA的权重$\lambda_1=0.8$、$\lambda_2=0.2$.从图中可以看出,算法LEA和算法BFTS的存储能力随着节点数量的增加在同步增强,而算法f-replica的存储能力随着节点数量的增加基本保持不变.这表明,基于纠删码的存储优化实现了区块链存储的横向扩展,即增加节点数量也会增强系统的存储能力.算法LEA和算法BFTS的存储能力提升得益于平均单区块存储代价的降低和稳定,由之前的分析可知,这两种算法的平均单区块存储代价会随着节点数量的增加在一定范围内波动.因为系统中所有节点的存储容量之和必定会随着节点数量的增加而增加,所以更多的节点意味着系统可以存储更多的区块,也即两种算法都实现了区块链存储的横向扩展.而算法f-replica的存储策略会产生较高的数据存储冗余,导致系统的存储能力不会随着节点数量的增加而增强,而是取决于系统中的瓶颈节点.如图5所示,算法LEA和算法BFTS在节点数量为5时,可以存储200多个区块,在节点数量为40时,两种算法已经可以存储1300多个区块;而算法f-replica在不同规模的网络中都只能存储300个区块左右.从图5还可以看出,在节点数量相同的情况下,算法LEA相比算法BFTS对区块链存储能力的提升更高.比如,当节点数量为25时,采用算法LEA可以存储867个区块,而采用算法BFTS可以存储850个区块.同时,算法LEA的存储能力显著高于算法f-replica,当网络中的节点数量为40时,算法LEA可以比算法f-replica多存储1080个区块.

图6(a)和图6(b)分别展现了节点对区块读取的概率为均匀分布和重尾分布时,各算法产生的平均延迟代价与节点数量之间的关系,其中恶意节点数量为0,算法LEA的权重$\lambda_1=0.8$、$\lambda_2=0.2$.随着节点数量的增加,算法LEA和算法BFTS的平均延迟代价总体呈上升趋势,而算法f-replica的平均延迟代价在小幅范围内稳定波动.这是因为算法LEA和算法BFTS在全网中只选择少量的节点保存编码数据块,同时这两种算法的平均单区块存储代价不会随着节点数量的增加而上升,所以当网络规模扩大时,算法LEA和算法BFTS的平均延迟代价总体呈上升趋势.而算法f-replica会在至少

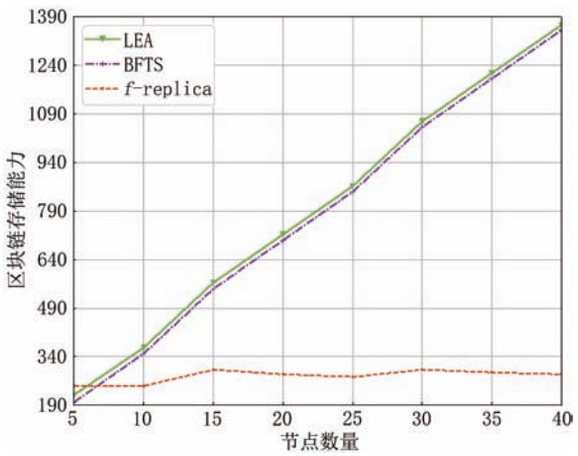

图5 节点数量对区块链存储能力的影响

$\lfloor(N-1)/3\rfloor+1$个节点上存储区块,即整个网络中将近1/3的节点会保存同一区块,所以当网络规模扩大时,算法f-replica的数据读取性能依然可以保持稳定.值得注意的是,在不同节点数量的情况下,算法LEA产生的平均延迟代价均小于算法BFTS.当采用均匀分布产生节点对区块的读取概率时,算法LEA的数据读取性能最好时优于算法BFTS 15.2%;在采用重尾分布产生节点对区块读取概率时,算法LEA相对于算法BFTS的性能提升最高可以达到13.4%.因为算法LEA在选择节点存储编码数据块时,考虑到了节点间的延迟以及节点对编码数据块的读取概率,所以会将编码数据块放置在对其具有高读取概率的区域.而算法BFTS仅考虑可靠性约束,未考虑节点间的延迟以及节点对编码数据块的读取概率.此外,当节点数量从5增加到40时,在均匀分布的情况下,算法LEA的平均延迟代价高于算法f-replica约3.4%-120.4%;在重尾分布的情况下,算法LEA的平均延迟代价要比算法f-replica高3.3%-110.2%,但是算法LEA的平均单区块存储代价最好时比算法f-replica低3.78倍(图4所示),也即算法LEA虽然会损失部分数据读取性能,但是能取得更好的存储性能,所以整体性能更优.

通过以上实验可以得知,当算法LEA的权重$\lambda_1=0.8$、$\lambda_2=0.2$时,算法LEA在存储代价和数据读取性能两方面都取得了比算法BFTS更好的效果;同时,算法LEA相比算法f-replica可以在存储代价和数据读取性能之间实现更好的平衡.

5.2.2 恶意节点数量对延迟代价的影响

本小节评估恶意节点数量对数据读取延迟的影响.算法LEA的权重$\lambda_1=0.8$、$\lambda_2=0.2$.图7评估的

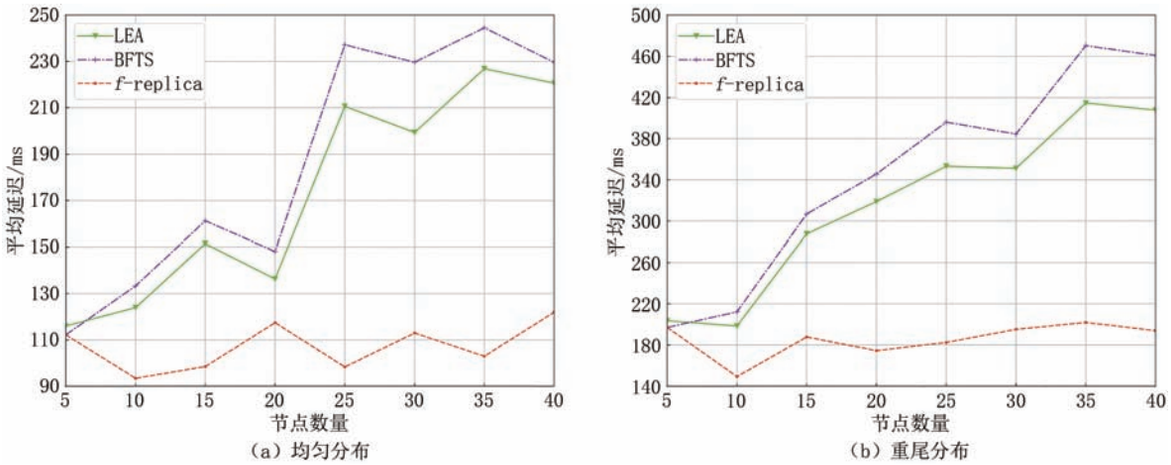

图6　节点数量对平均延迟代价的影响

是在40个节点的网络中，各算法产生的平均延迟代价与恶意节点数量的关系，图7(a)和图7(b)中节点对区块读取的概率分别为均匀分布和重尾分布. 从图7可以看出，各个算法的平均延迟代价随着恶意节点数量的增加而增大. 随着恶意节点数量的增加，节点从相距较远的节点处读取数据的概率以及进行纠删码解码的概率会上升，从而导致数据读取延迟的上升. 当恶意节点数量相同时，算法LEA相比算法BFTS可以取得较低的数据读取延迟. 例如，在均匀分布和重尾分布的情况下，算法LEA优于算法BFTS最好时分别可达15.4%和15.3%. 当恶意节点的数量从0增加到10时，在均匀分布和重尾分布的情况下，虽然算法LEA的平均延迟代价最坏时比算法f-replica分别要高99.6%和135.2%，但是算法LEA的平均单区块存储代价比算法f-replica低3.78倍；也即在恶意节点存在的情况下，算法LEA相比算法f-replica可以实现更好的存储代价和数据读取性能的平衡.

5.2.3　存储代价和延迟代价权重对算法的影响

本小节分析存储代价权重λ_1和延迟代价权重λ_2对算法LEA的影响，实验中设置$\lambda_1+\lambda_2=1$，节点数量为40，恶意节点数量为0，节点对数据的读取概率采用均匀分布产生. 图8和图9展示了在不同规模的网络中，算法LEA的平均单区块存储代价和平均延迟代价与不同存储代价权重的关系. 可以看到，当节点数量固定时，随着存储代价权重的降低，算法LEA的平均单区块存储代价在上升而平均延迟代价在下降. 因为随着存储代价权重的降低，存储代价对目标函数的影响下降，算法LEA会选择存储较多的编码数据块来降低平均读取延迟. 当存储代价权重固定时，算法LEA的平均延迟代价随着节点数量的增加总体呈上升趋势. 在实际中，可根据不同需求设置不同的权重值来获得最有利的编码数据块存储方案.

5.3　真实的联盟链系统中进行实验验证

为了进一步评估算法LEA在真实联盟链系统

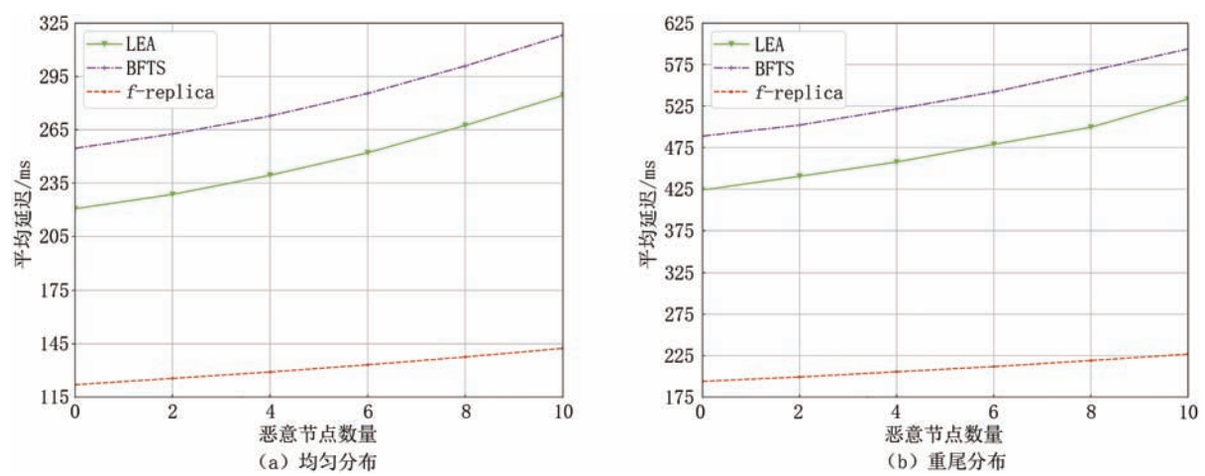

图7　40个节点的网络中恶意节点数量对平均延迟代价的影响

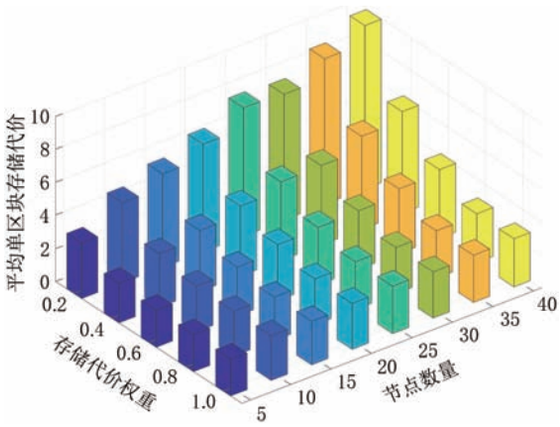

图8 存储代价权重对平均单区块存储代价的影响

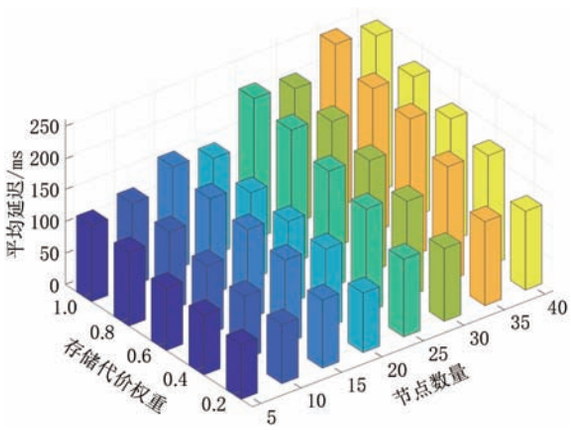

图9 存储代价权重对平均延迟代价的影响

中的性能，本节在Tendermint平台上进行了实验验证，所有的实验都是在5台机器组成的集群上完成的，网络带宽为100 Mbps，每个节点分配的存储空间为100 MB. 同时为了测试算法在不同规模网络下的性能，每台机器上会运行多个（最多8个）节点，以模拟一个最多拥有40个节点的中型网络[14]. 实验中使用的5台机器的硬件配置如表3所示.

表3 机器硬件配置表

机器序号	CPU	内存
1	4核，2.50 GHz	8 GB
2	4核，2.30 GHz	8 GB
3	4核，2.30 GHz	12 GB
4	8核，3.60 GHz	16 GB
5	12核，2.60 GHz	16 GB

5.3.1 节点数量对存储代价和延迟代价的影响

图10和图11分别展示了节点数量对各算法的平均单区块存储代价和系统存储能力的影响，算法LEA的权重$\lambda_1=0.8$、$\lambda_2=0.2$. 与仿真环境中的实验结果相比，各算法的平均单区块存储代价和系统存储能力的变化趋势均保持不变. 图10显示，当节点数量从5增加到40时，算法LEA的平均单区块存储代价最好时优于算法BFTS 11.1%. 此外，在不同规模的网络中，算法f-replica始终会选择$f+1$个节点存储区块，其中f等于$\lfloor (N-1)/3 \rfloor$，所以算法f-replica的平均单区块存储代价与仿真实验中得到的数据基本相同. 从图11可以看出，在节点数量相同的情况下，采用算法LEA的区块链系统可以存储更多的区块. 例如，当节点数量为40时，采用算法LEA可以存储1366个区块；采用算法BFTS可以存储1350个区块；而采用算法f-replica仅可以存储286个区块.

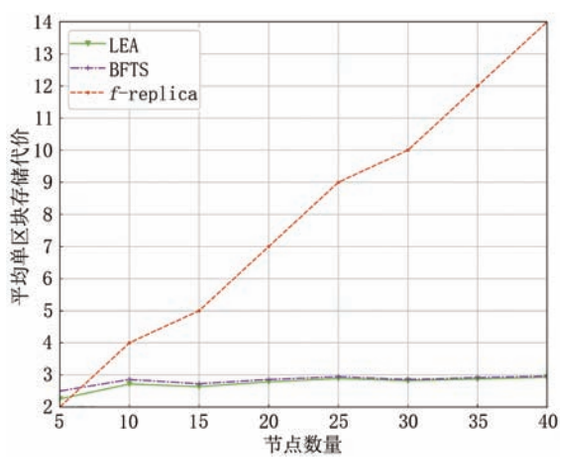

图10 节点数量对平均单区块存储代价的影响

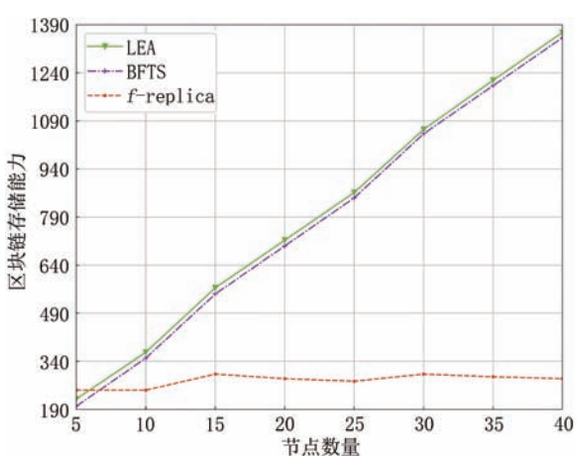

图11 节点数量对区块链存储能力的影响

图12(a)和图12(b)分别展现了节点对区块读取的概率为均匀分布和重尾分布时，各算法产生的平均延迟代价与节点数量之间的关系，其中恶意节点数量为0，算法LEA的权重$\lambda_1=0.8$、$\lambda_2=0.2$. 从图12可以看出，随着节点数量的增加，三种

算法的平均延迟代价都在增高，这是因为实验中的单台机器上运行了多个节点，节点之间的通信延迟会随着网络规模的扩大而增加，从而导致各算法的平均延迟代价也在增加.在不同节点数量的情况下，算法LEA产生的平均延迟代价低于算法BFTS.当采用均匀分布产生节点对区块的读取概率时，算法LEA的数据读取性能最高优于算法BFTS 14.6%；在采用重尾分布产生节点对区块读取概率时，LEA相对于算法BFTS的性能提升最高可以达到11.9%.从图12还可以看出，当节点数量从5增加到40时，在均匀分布的情况下，算法LEA的平均延迟代价高于算法f-replica为3.9%-52.9%；在重尾分布的情况下，算法LEA的平均延迟代价要比算法f-replica高7.3%-100.1%，但是算法LEA的平均单区块存储代价最好时比算法f-replica低3.78倍.

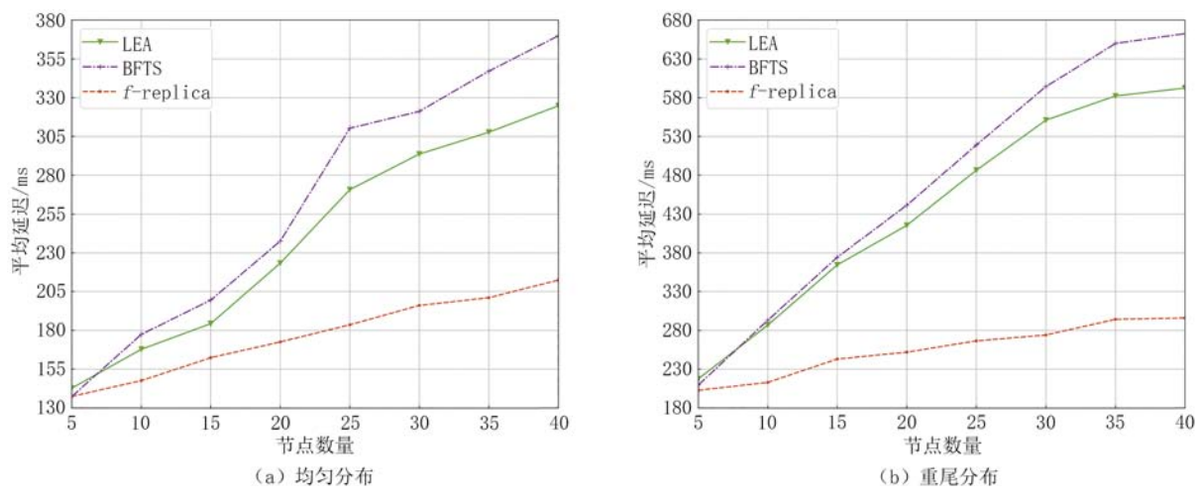

图12　节点数量对平均延迟代价的影响

5.3.2　恶意节点数量对延迟代价的影响

本小节评估恶意节点数量对数据读取延迟的影响.算法LEA的权重$\lambda_1=0.8$、$\lambda_2=0.2$.图13衡量的是在40个节点的网络中，算法LEA、算法BFTS以及算法f-replica产生的平均延迟代价与恶意节点数量的关系，图13(a)和图13(b)中节点对区块读取的概率分别为均匀分布和重尾分布.从图13可以看出，各算法的平均延迟代价随着恶意节点数量的增加而增高.当恶意节点数量相同时，算法LEA相比算法BFTS可以取得较低的数据读取延迟.例如，在均匀分布和重尾分布的情况下，算法LEA优于算法BFTS最好时分别可达13.9%和13.3%.同时，当恶意节点的数量从0增加到10时，在均匀分布和重尾分布的情况下，算法LEA的平均延迟代价最坏时比算法f-replica分别要高72.9%和116.3%，但是算法LEA的平均单区块存储代价比算法f-replica低3.78倍.

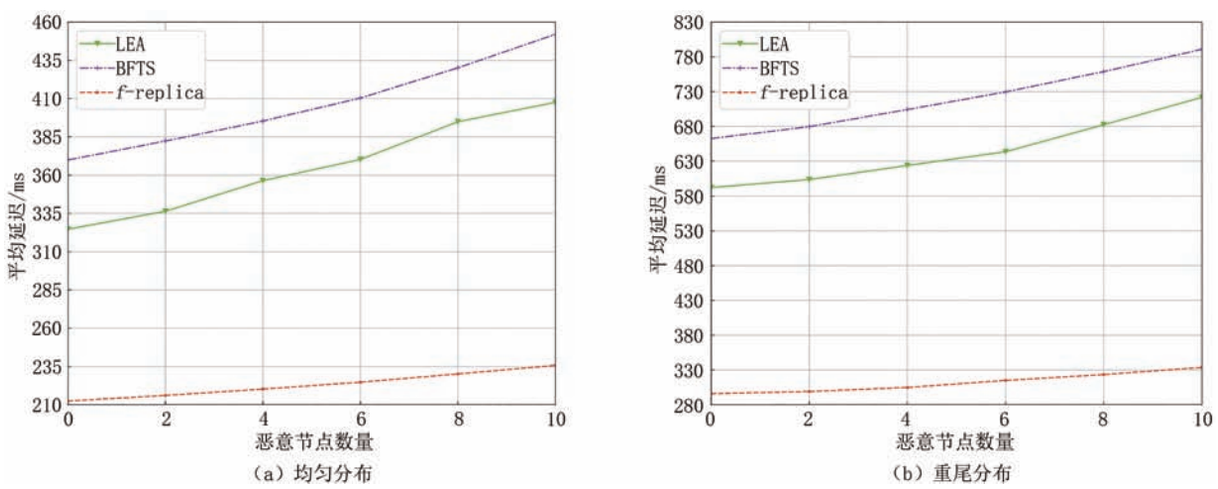

图13　40个节点的网络中恶意节点数量对平均延迟代价的影响

5.3.3 存储代价和延迟代价权重对算法的影响

本小节分析存储代价权重λ_1和延迟代价权重λ_2对算法LEA的影响，实验中设置$\lambda_1+\lambda_2=1$，节点数量为40，恶意节点数量为0，节点对数据的读取概率采用均匀分布产生.从图14和图15可以看到，当节点数量固定时，随着存储代价权重的降低，算法LEA的平均单区块存储代价在上升而平均延迟代价在下降.当存储代价权重固定时，随着节点数量的增加，算法LEA的平均延迟代价在增大，这是因为在5台机器组成的集群中，区块链节点数量的增加会导致节点之间通信延迟增大，进而导致算法LEA的平均延迟代价增大.

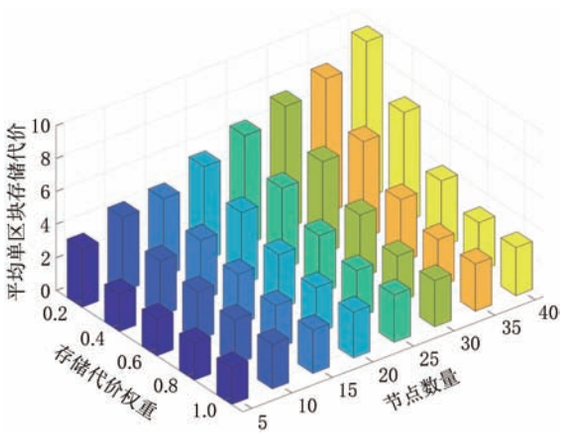

图14 存储代价权重对平均单区块存储代价的影响

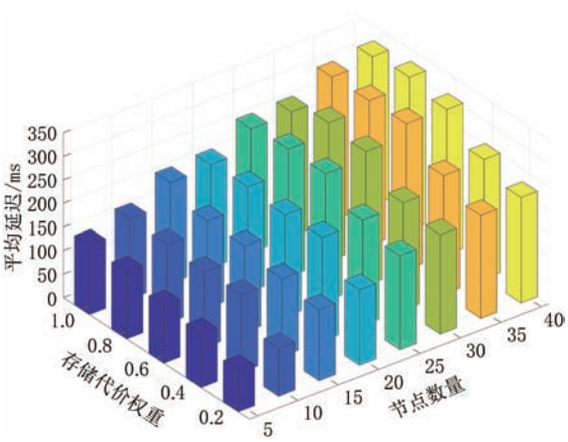

图15 存储代价权重对平均延迟代价的影响

通过在真实的联盟链系统中进行实验验证，可以得知算法LEA在不同规模的网络中以及不同恶意节点数量的网络中都能比算法BFTS取得更好的数据存储和数据读取性能.同时，相比算法f-replica，算法LEA可以有效降低存储冗余，并在存储代价和数据读取性能之间实现更好的平衡.与仿真环境下的实验结果相比，真实环境下得到的实验数据总体变化趋势保持不变，从而进一步说明了算法LEA性能的高效和稳定.

6 总 结

区块链要求每个节点都存储一份完整的数据，导致整个系统的存储能力不具有横向扩展性，进而限制了区块链的发展.采用纠删码存储区块中的数据，可以有效地减少存储冗余.编码数据块的存储数量及存储位置是影响数据的可靠性、存储代价和数据读取延迟的关键因素.本文在基于纠删码的BFT联盟链中，研究编码数据块的存储数量及存储位置决策问题，以在满足数据可靠性的约束下获得平衡的数据存储和读取性能.针对编码数据块的存储数量及存储位置决策问题，提出了延迟感知的编码数据块分配算法LEA.算法LEA首先求解编码数据块的存储数量及存储位置决策问题的松弛问题以及该松弛问题的对偶问题，然后根据松弛问题及其对偶问题的最优解依次为每个编码数据块确定其存储数量和存储位置，最后调整得到的编码数据块存储分配方案使其满足被松弛的约束条件.理论分析证明，算法LEA是$\ln 3+2$近似算法.通过在仿真环境和真实的联盟链系统中进行实验，可以得知算法LEA有效降低了区块链系统的存储冗余，从而使得系统的存储能力随着节点数量的增加而增强，实现了系统存储的横向扩展.同时，算法LEA在选择节点存储编码数据块时，考虑到了节点间的延迟以及节点对编码数据块的读取概率，所以在不同规模大小的网络中和不同恶意节点数量的网络中，算法LEA相比算法BFTS都取得了较好的数据读取性能，性能提升最好时分别达到15.2%和15.4%.与算法f-replica相比，算法LEA虽然会损失少部分数据读取性能，但是能够取得更好的存储性能，从而使得整体性能更优.此外，在不同规模的网络中，随着存储代价权重的降低，算法LEA的平均单区块存储代价在上升而平均延迟代价在下降，也即算法LEA在存储代价和数据读取性能之间实现了良好的平衡.在实际中，可以根据不同的需求设置不同的权重值来获得最有利的编码数据块存储方案.

参 考 文 献

[1] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system. White Paper, 2008

[2] Fu X, Wang H, Shi P. A survey of blockchain consensus

algorithms：mechanism，design and applications. Science China Information Sciences，2021，64(2)：1-15

[3] Yu Ge，Nie Tie-Zheng，Li Xiao-Hua，et al. The challenge and prospect of distributed data management techniques in blockchain systems. Chinese Journal of Computers，2021，44(1)：28-54 (in Chinese)
(于戈，聂铁铮，李晓华等. 区块链系统中的分布式数据管理技术——挑战与展望. 计算机学报，2021，44(1)：28-54)

[4] Arenas R，Fernandez P. CredenceLedger：A permissioned blockchain for verifiable academic credentials//Proceedings of the 2018 IEEE International Conference on Engineering，Technology and Innovation. Stuttgart，Germany，2018：1-6

[5] Cheng X，Chen F，Xie D，et al. Blockchain-based secure authentication scheme for medical data sharing//Proceedings of the 5th International Conference of Pioneering Computer Scientists，Engineers and Educators. Guilin，China，2019：396-411

[6] Yao Z，Pan H，Si X，et al. Decentralized access control encryption in public blockchain//Proceedings of the International Conference on Blockchain and Trustworthy Systems. Guangzhou，China，2019：240-257

[7] Abdellatif K，Abdelmouttalib C. Graph-based computing resource allocation for mobile blockchain//Proceedings of the 2018 6th International Conference on Wireless Networks and Mobile Communications. Marrakesh，Morocco，2018：1-4

[8] Sun Zhi-Xin，Zhang Xin，Xiang Feng，et al. Survey of storage scalability on blockchain. Journal of Software，2021，32(1)：1-20 (in Chinese)
(孙知信，张鑫，相峰等. 区块链存储可扩展性研究进展. 软件学报，2021，32(1)：1-20)

[9] Shao Qi-Feng，Jin Che-Qing，Zhang Zhao，et al. Blockchain：Architecture and research progress. Chinese Journal of Computers，2018，41(5)：969-988 (in Chinese)
邵奇峰，金澈清，张召等. 区块链技术：架构及进展. 计算机学报，2018，41(5)：969-988

[10] Jia Da-Yu，Xin Jun-Chang，Wang Zhi-Qiong，et al. Efficient query model for storage capacity scalable blockchain system. Journal of Software，2019，30(9)：2655-2670 (in Chinese)
贾大宇，信俊昌，王之琼等. 存储容量可扩展区块链系统的高效查询模型. 软件学报，2019，30(9)：2655-2670

[11] Kadhe S，Chung J，Ramchandran K. SeF：A secure fountain architecture for slashing storage costs in blockchains. arXiv preprint arXiv：1906. 12140，2019

[12] Zhu Yu-Jin，Yao Jian-Guo，Guan Hai-Bing. Blockchain as a service：Next generation of cloud services. Journal of Software，2019，31(1)：1-19 (in Chinese)
(朱昱锦，姚建国，管海兵. 区块链即服务：下一个云服务前沿. 软件学报，2019，31(1)：1-19)

[13] Pan Chen，Liu Zhi-Qiang，Liu Zhen，et al. Research on scalability of blockchain technology：problems and methods. Journal of Computer Research and Development，2018，55(10)：2099-2110 (in Chinese)
(潘晨，刘志强，刘振等. 区块链可扩展性研究：问题与方法. 计算机研究与发展，2018，55(10)：2099-2110)

[14] Qi X，Zhang Z，Jin C，et al. BFT-Store：Storage partition for permissioned blockchain via erasure coding//Proceedings of the 2020 IEEE 36th International Conference on Data Engineering. Dallas，USA，2020：1926-1929

[15] Reed I S，Solomon G. Polynomial codes over certain finite fields. Journal of the Society for Industrial and Applied Mathematics，1960，8(2)：300-304

[16] Castro M，Liskov B. Practical byzantine fault tolerance//Proceedings of the Third USENIX Symposium on Operating Systems Design and Implementation. New Orleans，USA，1999：173-186

[17] Mitra D，Dolecek L. Patterned erasure correcting codes for low storage-overhead blockchain systems//Proceedings of the 2019 53rd Asilomar Conference on Signals，Systems，and Computers. Pacific Grove，USA，2019：1734-1738

[18] Zyskind G，Nathan O，Pentland A S. Decentralizing privacy：Using blockchain to protect personal data//Proceedings of the 2015 IEEE Security and Privacy Workshops. San Jose，USA，2015：180-184

[19] Xu Q，Song Z，Goh R S M，et al. Building an ethereum and IPFS-based decentralized social network system//Proceedings of the 2018 IEEE 24th International Conference on Parallel and Distributed Systems. Singapore，2018：1-6

[20] Ali M S，Dolui K，Antonelli F. IoT data privacy via blockchains and IPFS//Proceedings of the Seventh International Conference on the Internet of Things. New York，USA，2017：1-7

[21] He G，Su W，Gao S. Chameleon：A scalable and adaptive permissioned blockchain architecture//Proceedings of the 2018 1st IEEE International Conference on Hot Information-Centric Networking. Shenzhen，China，2018：87-93

[22] Dai M，Zhang S，Wang H，et al. A low storage room requirement framework for distributed ledger in blockchain. IEEE Access，2018，6：22970-22975

[23] Abe R，Suzuki S，Murai J. Mitigating bitcoin node storage size by DHT//Proceedings of the Asian Internet Engineering Conference. New York，USA，2018：17-23

[24] Zamani M，Movahedi M，Raykova M. Rapidchain：Scaling blockchain via full sharding//Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. New York，USA，2018：931-948

[25] Wang J，Wang H. Monoxide：Scale out blockchains with asynchronous consensus zones//Proceedings of the 16th USENIX Symposium on Networked Systems Design and Implementation. Boston，USA，2019：95-112

[26] Perard D，Lacan J，Bachy Y，et al. Erasure code-based low storage blockchain node//Proceedings of the 2018 IEEE International Conference on Internet of Things and IEEE Green Computing and Communications and IEEE Cyber，Physical and Social Computing and IEEE Smart Data. Halifax，Canada，2018：1622-1627

[27] Wu H，Ashikhmin A，Wang X，et al. Distributed error

correction coding scheme for low storage blockchain systems. IEEE Internet of Things Journal, 2020, 7(8): 7054-7071

[28] Raman R K, Varshney L R. Dynamic distributed storage for blockchains//Proceedings of the 2018 IEEE International Symposium on Information Theory. Vail, USA, 2018: 2619-2623

[29] Zhao Guo-Feng, Zhang Ming-Cong, Zhou Ji-Hua, et al. Research and application of block file storage model based on blockchain system of erasure code. Netinfo Security, 2019, 19(2): 28-35 (in Chinese)
(赵国锋，张明聪，周继华等. 基于纠删码的区块链系统区块文件存储模型的研究与应用. 信息网络安全，2019，19(2)：28-35)

[30] Pal R. Fountain coding for bootstrapping of the blockchain// Proceedings of the 2020 International Conference on Communication Systems and Networks. Bengaluru, India, 2020: 1-5

[31] Qi X, Zhang Z, Jin C, et al. A reliable storage partition for permissioned blockchain. IEEE Transactions on Knowledge and Data Engineering, 2021, 33(1): 14-27

[32] Wang Yi-Jie, Xu Fang-Liang, Pei Xiao Qiang. Research on erasure code-based fault-tolerant technology for distributed storage. Chinese Journal of Computers, 2017, 40(1): 236-255 (in Chinese)
(王意洁，许方亮，裴晓强. 分布式存储中的纠删码容错技术研究. 计算机学报，2017，40(1)：236-255)

[33] Wang Zi-Zhong, Wang Hai-Xia, Shao Ai-Ran, et al. A local-reconstruction-code-and-hitchhiker-code-mixing storage scheme. Chinese Journal of Computers, 2020, 43(4): 618-630 (in Chinese)
(王梓仲，王海霞，邵艾然等. 一种混合局部恢复码及Hitchhiker码的存储策略. 计算机学报，2020，43(4)：618-630)

[34] Chudak F A, Shmoys D B. Improved approximation algorithms for the uncapacitated facility location problem. SIAM Journal on Computing, 2003, 33(1): 1-25

FAN Yu-Qi, Ph. D., associate professor. His research includes blockchain, computer networks, cloud computing, artificial intelligence, etc.

SHENG Dong, M. S. candidate. His research is blockchain.

WANG Lun-Fei, M. S. candidate. His research includes blockchain and computer networks.

Background

This work is supported by the National Key Research and Development Program of China (2018YFB2000505), the National Natural Science Foundation of China (61806067), and the Key Research and Development Program of Anhui Province, China (201904a06020024). Blockchain requires each node to store a complete copy of data to ensure data reliability with high storage redundancy. However, it also brings huge storage pressure to blockchain nodes and reduces the utilization of storage resources, which makes the storage scalability be a bottleneck of blockchain. Using erasure code to encode the data stored in the blockchain can effectively reduce the storage redundancy. Nevertheless, the reduction of storage redundancy will reduce the reliability of the data, incur data recovery costs, and increase the data access delay. Currently, the work on reducing the storage redundancy in blockchains ignores the impact of inter-node delay and encoded data chunk storage locations on data reliability and data access delay during the data block placements.

This paper summarizes the previous related studies on storage redundancy reduction in blockchains. Specifically, this paper investigates the decision problem of the number and locations of encoded data chunk copies with the erasure code in BFT permissioned blockchains, with the aim to achieve a good balance between data storage cost and data access latency under the data reliability constraint. We also propose a Latency-aware Encoded data chunks Allocation algorithm (LEA). Firstly, algorithm LEA solves the relaxation problem of the studied problem and the dual problem of the relaxation problem. Algorithm LEA then determines the number and locations of data chunk copies to be stored according to the optimal solution of the relaxation problem and its dual problem. Finally, the algorithm adjusts the storage allocation solution to satisfy the constraints of the decision problem. Theoretical analysis proves that algorithm LEA is an approximation algorithm of $\ln 3+2$ to the data allocation decision problem. The experimental results show that algorithm LEA can achieve good performance in terms of the tradeoff between data storage cost and data access delay.