分布式矩阵–向量乘掉队节点编码新方案

韦宝典，莫肇豪，马啸
（中山大学计算机学院，广东 广州 510006）

摘　要：编码分布式计算（CDC）是利用纠删码解决掉队节点问题的一种主流方案，但是现有相关工作忽略了掉队节点的计算性能，产生了较多额外编译码时间。针对此问题，基于阶束矩阵码（HBMC），提出固定码率阶束编码分布式计算（F-HB）以及无速率阶束编码分布式计算（R-HB）两类方案，其中，F-HB 能够解决掉队节点问题，降低编译码时间；R-HB 利用掉队节点完成计算。理论分析和实验仿真均证明了所提方案在解决掉队节点问题、降低额外编译码时间及提升计算效率方面的有效性。仿真结果表明，与未编码分布式计算（UDC）方案相比，所提两类方案任务时间分别减少 68% 与 74%。基于 HBMC 的 F-HB 和 R-HB 方案通过降低编译码时间、利用掉队节点计算性能，显著缩短了分布式矩阵–向量乘计算系统的任务时间，为解决掉队节点问题提供了高效可行的新途径。

关键词：分布式计算；矩阵–向量乘；阶束矩阵码；固定码率阶束编码分布式计算；无速率阶束编码分布式计算
中图分类号：TN919.3+1
文献标志码：A
DOI：10.11959/j.issn.1000–436x.2026027

New coding scheme for straggler mitigation in distributed matrix-vector multiplication

Wei Baodian, Mo Zhaohao, Ma Xiao
School of Computer and Engineering, Sun Yat-sen University, Guangzhou 510006, China

Abstract: Coded distributed computing (CDC) has been recognized as a classic method to mitigate stragglers via erasure codes. Nevertheless, previous studies ignored the work completed by stragglers and incurred additional coding overheads. Thus, two coded computation methods based on hierarchical bundle matrix codes (HBMC) were proposed, fixed-rate hierarchical bundle coded distributed computing (F-HB) and rateless hierarchical bundle coded distributed computing (R-HB). F-HB was designed to solve the straggler problem and reduce coding latency. Based on this, R-HB further exploited the work completed by stragglers. The effectiveness of the proposed schemes was verified through theoretical analysis and experimental simulation. Numerical results show that F-HB and R-HB reduce task time by approximately 68% and 74% respectively, compared to uncoded distributed computing (UDC) methods. F-HB and R-HB based on HBMC reduce the task time of distributed matrix-vector multiplication computing systems significantly by lowering the latency of coding and exploiting the work completed by stragglers. The proposed schemes provide an efficient and feasible new approach to solving the straggler problem in distributed computing systems.

Keywords: distributed computation, matrix-vector multiplication, hierarchical bundle matrix code, F-HB, R-HB

收稿日期：2025–11–17；**修回日期：**2026–01–27
通信作者：韦宝典，weibd@mail.sysu.edu.cn
基金项目：国家自然科学基金资助项目（No.62471506）；国家重点研发计划基金资助项目（No.2021YFA1000500）
Foundation Items: The National Natural Science Foundation of China (No.62471506), The National Key Research and Development Program of China (No.2021YFA1000500)

0　引言

在大数据人工智能时代，机器学习、数据挖掘等应用会涉及大规模的矩阵和向量乘计算。要完成此类计算任务，需要采用更高性能的计算系统，例如分布式计算系统。分布式计算是指由一组计算设备（称为计算节点）组成的系统模型。节点之间会共享网络和存储资源，共同完成任务。相较过去的集中式计算，分布式计算的任务负载由多个节点承担，因而能提供高效的计算服务。由于分布式系统的实现成本低，容错性能高，分布式计算已应用在网络服务、并行计算等领域中[1]。

近几年，国内外学者对分布式计算相关问题进行了深入的研究[2-6]，例如掉队节点问题。在完成任务的过程中，部分节点可能出现掉队现象，即这些节点完成任务的速度远低于其他节点，或者由于争夺共享资源、能源不足、网络拥塞等问题暂时断开与网络的连接[7]。掉队节点会显著增加系统的任务时间。

解决掉队节点问题的朴素做法大致有3类：一是将低速节点任务重新分配给高速节点[8]，但该做法未能充分利用低速节点的计算性能；二是在某个节点完成任务后，系统重新分配当前所有未完成的任务[9]，但该做法的通信开销大；三是简单地复制同一子任务给多个节点进行计算，结果由计算速度最快的节点确定[10]，但该做法同样会造成高通信负载和计算开销。

1　相关工作

近年来，编码分布式计算（coded distributed computing，CDC）已成为解决掉队节点问题的主流方案。CDC将掉队节点问题视作一种删除信道模型，并通过纠删码恢复掉队节点信息[11]。相较前述方法，CDC产生的时延和开销更低。

文献[11]将极大距离可分（maximum distance separable，MDS）码应用到计算线性任务的分布式系统中。该方案将掉队节点的计算结果视作被删除的数据，并通过其他节点的计算结果加以恢复。基于重复编码与MDS编码，文献[12]设计了一种近似计算方案，通过划分子任务的优先级，减轻掉队节点的影响。文献[13]提出了一种稀疏编码方法，通过随机加权组合和混合解码策略，实现了接近最优的恢复阈值和线性解码时间。文献[14]提出一种低权重（low-weight，LW）编码的方法，把计算矩阵分割为多个子矩阵块，并将少量子块的随机线性叠加分配给每个工作节点，以减少编码后矩阵的非零元素数量，降低计算开销。文献[15]采用卢比变换（Luby transform，LT）码构造生成矩阵，并使用高效译码算法进行译码，降低译码复杂度。在此基础上，文献[16]在LT码译码的过程中加入了反馈机制，使主节点能够利用反馈信息，进一步加快译码速度。文献[17]提出了一种基于空中计算和模拟编码的方法，通过任务粒度调整和无线资源协同，缓解掉队问题。文献[18]提出了一种基于代数函数域的编码方法。该方法利用代数曲线上的函数域而非传统多项式，在固定有限域上构造编码，从而将问题转化为代数曲线的半群组合优化问题，实现接近最优的恢复阈值。上述方案通过引入冗余，解决掉队节点问题。然而，上述方案仅通过正常节点的计算结果恢复丢失信息，完全舍弃了掉队节点的计算性能。当系统中掉队节点较多时，计算资源会被大量浪费。

针对计算资源浪费问题，可以进一步细分计算任务，使掉队节点在有限的计算速度下能够完成少量任务。文献[19]提出了贝鲁特（Berrut）近似编码计算方法，基于Berrut有理插值进行编解码，能够近似计算任意函数，且不需要固定恢复阈值。文献[20]提出了草图编码方法，通过结合草图计数随机压缩技术与结构化多项式编码，能够在保证近似精度的同时，显著减少需要等待的节点数量。文献[21]提出一种具有层结构的CDC方案。该方案将计算矩阵划分为多个子计算层，节点完成当前层计算任务后，才会计算下一层任务。该方案采用不同码率的MDS码对不同层进行编码，使掉队节点参与部分任务的计算。文献[22]将高维矩阵乘法任务分解为多个子任务，并通过多项式编码（如拉格朗日多项式）生成冗余计算任务，以容忍掉队节点。该方案会根据节点计算性能进行动态任务分配，并回收所有已完成的计算来恢复最终结果，避免资源浪费。文献[23]提出无速率LT码CDC方案，利用LT码能够生成任意数量编码符号，且接收足够数量编码符号即可译码的特性，将掉队节点的计算结果也纳入译码过程中。文献[24]指出无速率LT码CDC方案所需额外编译码开销大，并提出一种基于猛禽码的无速率CDC优化方案，降低编译码开销和计

算时间。上述方案一定程度上解决了计算资源的浪费问题，但存在编译码复杂度较高的问题，导致额外的性能开销。

针对现有CDC方案中存在的问题，本文提出两种基于阶束矩阵码（hierarchical bundle matrix codes，HBMC）的CDC方案，以解决分布式矩阵–向量乘计算中的掉队节点问题。首先，提出固定码率阶束编码分布式计算（fixed-rate hierarchical bundle coded distributed computing，F-HB）方案，对计算矩阵进行编码，并将编码矩阵分发给各节点。当完成任务的节点个数达到恢复阈值时，主节点就能够高概率恢复目标结果。其次，提出无速率阶束编码分布式计算（rateless hierarchical bundle coded distributed computing，R-HB）方案，利用HBMC的阶束结构，选取随机的束对计算矩阵进行编码，产生编码符号并不断发送给当前空闲节点，直到主节点成功恢复目标结果。本文的核心贡献如下。

1) 提出基于HBMC的两类CDC方案（F-HB和R-HB），分别解决固定码率场景下的低开销译码问题和无速率场景下的掉队节点计算利用问题，填补了现有方案在“低复杂度+资源高效利用”方面的空白。

2) 建立了F-HB和R-HB的期望任务时间模型，通过理论推导证明了方案在时延性能上的近最优性，为方案参数优化提供了理论依据。

3) 与MDS码、LT码、LW码等传统及新近方案进行仿真实验对比，验证了本文方案在任务时间、鲁棒性方面的优势。

4) 所提方案可直接推广至傅里叶变换、梯度下降等非线性分布式计算场景，具有广泛的应用价值。

2　问题建模

考虑由一个主节点和k个计算节点构成的分布式系统，如图1所示。

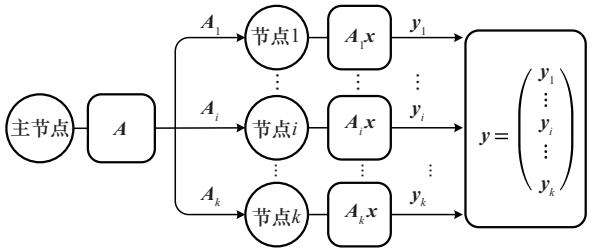


图1　未编码的分布式计算系统

系统的任务是计算矩阵–向量乘$\boldsymbol{y}=\boldsymbol{Ax}$，其中$\boldsymbol{A}\in\mathbb{F}^{m\times n}$，$\boldsymbol{x}\in\mathbb{F}^{n}$，而$\mathbb{F}$为有限域。为此，主节点将矩阵$\boldsymbol{A}$按行尽可能地均分为$k$个子矩阵，即$\boldsymbol{A}=\left(\boldsymbol{A}_1^{\mathrm{T}},\boldsymbol{A}_2^{\mathrm{T}}\cdots,\boldsymbol{A}_k^{\mathrm{T}}\right)^{\mathrm{T}}$，并将$\boldsymbol{A}_i$和$\boldsymbol{x}$发送给第$i$个节点，$i\in\{1,\cdots,k\}$；第$i$个节点在本地计算子任务$\boldsymbol{y}_i=\boldsymbol{A}_i\boldsymbol{x}$，然后将结果$\boldsymbol{y}_i$返回主节点。主节点收到所有$\boldsymbol{y}_i$后，即可得到$\boldsymbol{y}=\left(\boldsymbol{y}_1^{\mathrm{T}},\cdots,\boldsymbol{y}_k^{\mathrm{T}}\right)^{\mathrm{T}}$。该系统存在“掉队”节点，即部分节点的计算结果到达时间要远大于其他节点，甚至是无限大的[7]。由于系统的计算时间取决于最慢节点的计算时间，这些“掉队”节点会显著影响系统性能。

如图2所示，编码分布式计算CDC使用纠删码解决掉队节点问题。

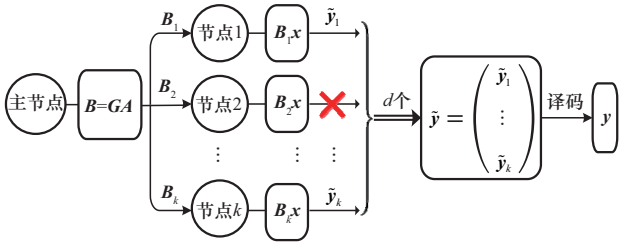


图2　采用纠删码编码的分布式计算系统

设一纠删码的生成矩阵为$\boldsymbol{G}\in\mathbb{F}^{kw\times m}$，其中$w>\dfrac{m}{k}$对应各节点所需计算的向量乘次数。主节点把$\boldsymbol{B}=\boldsymbol{GA}$按行均分给$k$个节点；第$i$个节点收到其对应的子矩阵$\boldsymbol{B}_i$后，计算并返回结果$\tilde{\boldsymbol{y}}_i=\boldsymbol{B}_i\boldsymbol{x}$。若主节点可以从任意$d$个节点返回的计算结果译码恢复$\boldsymbol{y}=\boldsymbol{Ax}$，则称$d$是可解节点数，称最小的可解节点数$\tilde{d}$是恢复阈值。此时要求由$\tilde{d}$个节点的子矩阵$\mathbf{B}_i$构成的矩阵集合中存在满秩矩阵。

3　基于HBMC的CDC方案

本节首先简单介绍HBMC，然后展示如何将HBMC应用到分布式矩阵–向量乘计算中，并在上述模型框架下提出两种基于HBMC的CDC方案，以解决掉队节点问题。

3.1　阶束矩阵码

HBMC是一种基于随机线性组合与分块思想的线性时间纠删码[25]，能够以高概率从任意$(1+\epsilon)m$个编码符号译码恢复出m个信息符号，其中ϵ为额外译码开销系数。HBMC具有良好的稀疏性，其编

译码复杂度低，仅为 $\mathcal{O}(\epsilon m^2)$。与 LT 码类似，HBMC 既能够根据固定码率 c，将 m 个信息符号编码为 cm 个编码符号，也能够以不固定的码率生成任意数量的编码符号。

考虑固定码率 HBMC，将信息序列记作 $\boldsymbol{U}$，生成矩阵记作 $\boldsymbol{G}$，其中 $\boldsymbol{U}\in\mathbb{F}^m$，$\boldsymbol{G}\in\mathbb{F}^{cm\times m}$。发送方先对 $\boldsymbol{U}$ 编码，得到码字序列 $\boldsymbol{V}=\boldsymbol{GU}$。$\boldsymbol{V}$ 在删除信道中传输，接收端通常情况下仅能收到 $\boldsymbol{V}$ 的一部分编码符号。在收到 $\boldsymbol{V}$ 中的任意 $(1+\epsilon)m$ 个编码符号后，通过 HBMC 译码算法[25]，接收端能够以 $1-\mathrm{e}^{-\Omega(\epsilon^2 m)}$ 的概率恢复出 $\boldsymbol{U}$，其中 Ω 表示下界。

HBMC 的生成矩阵 $\boldsymbol{G}$ 是根据特殊的规则进行构造的[26]。如图 3 所示，$\boldsymbol{G}$ 由多个随机生成的矩阵 $\boldsymbol{G}_{h'}$ 按阶排列而成，称 $\boldsymbol{G}_{h'}$ 为 HBMC 的束。$\boldsymbol{G}$ 中共有 h 阶不同的束，其中 $h=\log\left(\frac{1}{\epsilon^2}\right)$，$h'\in\{1,\cdots,h\}$。给定度分布 $\omega_{h'}$，满足 $\sum_{h'}\omega_{h'}=1+\epsilon$，各阶的 $\boldsymbol{G}_{h'}\in\mathbb{F}^{\frac{\omega_{h'}}{1+\epsilon}cm_{h'}\times m_{h'}}$，$m_{h'}=\frac{m}{2^{h-h'}}$。除 $\boldsymbol{G}_1$ 外，每个 $\boldsymbol{G}_{h'}$ 之上都有两个相同大小的 $\boldsymbol{G}_{h'-1}$，易知 $m_1=\frac{m}{2^{h-1}}=2\epsilon^2 m$，$m_h=m$。这种结构被称为阶束结构。阶束结构既能够通过度分布的设计提高译码成功概率，也能够保证 $\boldsymbol{G}$ 具有良好的稀疏性，降低编译码复杂度。

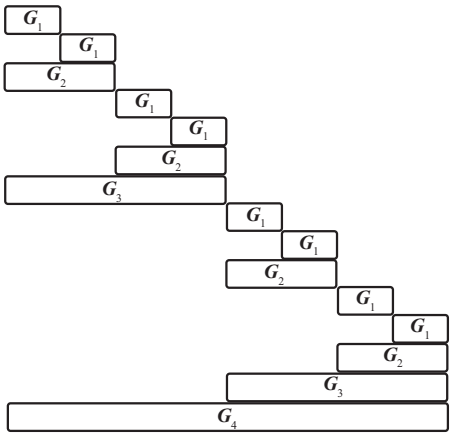


图 3　HBMC 生成矩阵 $\boldsymbol{G}(h=4)$

掉队节点的本质是“部分节点计算结果延迟或缺失，但仍可能完成部分计算任务”，对应删除信道中“符号部分丢失”的特性。而 HBMC 的阶束结构可以恢复部分丢失符号——其生成矩阵 $\boldsymbol{G}$ 由多阶束 $\boldsymbol{G}_{h'}$ 组成，低阶束（h' 较小）对应细粒度子任务，高阶束（h' 较大）对应聚合任务，即使部分低阶束计算结果缺失，主节点仍可通过高阶束的部分计算结果补充译码，不需要完全舍弃掉队节点的计算贡献。

3.2　F-HB 方案

F-HB 的主节点依据 3.1 节所述构造规则，构造固定码率的 HBMC 生成矩阵 $\boldsymbol{G}$，并计算 $\boldsymbol{B}=\boldsymbol{GA}$。可以将 $\boldsymbol{A}$ 视作由 n 列信息序列构造的矩阵 $(\boldsymbol{U}_1,\boldsymbol{U}_2,\cdots,\boldsymbol{U}_n)$，将 $\boldsymbol{B}$ 视作 $(\boldsymbol{V}_1,\boldsymbol{V}_2,\cdots,\boldsymbol{V}_n)$，将节点掉队视作删除部分行。主节点如图 2 所示进行任务分发。在收到任意 $\tilde{d}$ 个节点返回的计算结果后，主节点通过 HBMC 译码算法，能够以 $1-\mathrm{e}^{-\Omega(\epsilon^2 m)}$ 的概率译码得到 $\boldsymbol{y}=\boldsymbol{Ax}$。

相较先前固定码率 CDC 方案，在译码成功概率和恢复阈值相近的条件下，F-HB 的编译码复杂度更低。

3.3　R-HB 方案

随着系统中掉队节点数的增加，固定码率 CDC 方案会造成更多计算资源的浪费。基于 HBMC 的无速率特性，本文提出一种无速率 CDC 方案——R-HB 来解决该问题。

如图 4 所示，R-HB 的主节点产生编码符号 $\boldsymbol{B}_j=\boldsymbol{G}_j\boldsymbol{A}$ 并发送给当前的空闲节点，其中 $\boldsymbol{G}_j$ 为 $\boldsymbol{G}$ 的任意子行。第 i 个节点收到 $\boldsymbol{B}_j$ 后，会在本地计算 $\tilde{\boldsymbol{y}}_{i,j}=\boldsymbol{B}_j\boldsymbol{x}$，并将结果返回主节点，同时转为空闲状态，等待下一次任务。在收到任意 $(1+\epsilon)m$ 个返回的计算结果后，主节点通过 HBMC 译码算法，能够以 $1-\mathrm{e}^{-\Omega(\epsilon^2 m)}$ 的概率译码得到 $\boldsymbol{y}=\boldsymbol{Ax}$。

在实现低编译码复杂度的基础上，R-HB 通过任务细分的方式，利用掉队节点参与部分计算，从而改善计算资源浪费问题。

4　理论分析

本节对比不同编码方案的编译码复杂度和译码成功概率，然后，从理论角度分析 F-HB 与 R-HB 的期望任务时间。

4.1　编译码复杂度和译码成功概率

参考文献[26]中的图 1，以及文献[27]中的定理 13 和定理 17，得出编译码复杂度与译码成功概率对比如表 1 所示。需要注意的是，在 m 固定的条件下，

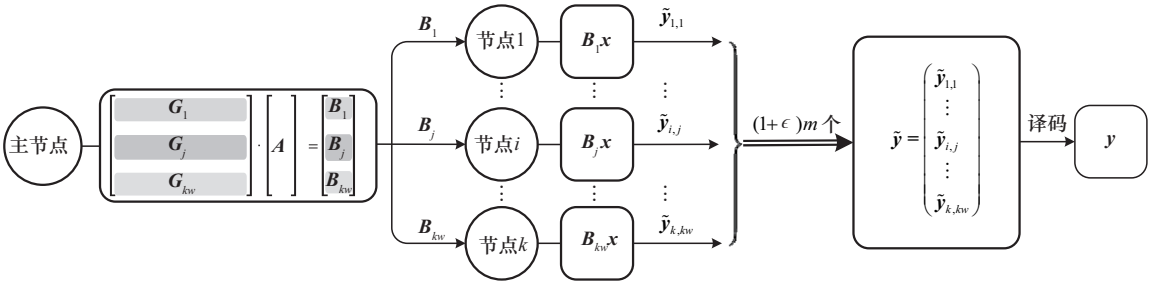


图4 采用无速率HBMC编码的分布式计算系统

为了确保HBMC具有高译码成功概率（一般达到0.95以上），需要调整ϵ，使其在满足$1-\mathrm{e}^{-\Omega(\epsilon^2 m)}\geqslant 0.95$条件下尽可能小。

表1 编译码复杂度与译码成功概率对比

编码方案	MDS码	LT码	HBMC
信息序列长	m	m	m
码字序列长	cm	cm	cm
所需符号数	m	$m+\mathcal{O}\left(\sqrt{m}\log^2(\frac{m}{\delta})\right)$	$(1+\epsilon)m$
译码成功概率	1	$1-\delta$	$1-\mathrm{e}^{-\Omega(\epsilon^2 m)}$
编码时间复杂度	$\mathcal{O}(m^2)$	$\mathcal{O}(m\log(n))$	$\mathcal{O}(\epsilon m^2)$
译码时间复杂度	$\mathcal{O}(m^2)$	$\mathcal{O}(m\log(m))$	$\mathcal{O}(\epsilon m^2)$

从表1可以看出，在确保一定译码成功概率的条件下，随着m增大，HBMC所需的编码冗余系数ϵ可以减小，复杂度也会随之降低。具体地，当$\epsilon\ll\frac{\log(m)}{m}$且$\epsilon\ll\frac{\log(n)}{m}$时，HBMC的编译码复杂度可以低于MDS码与LT码。在实际的大规模矩阵-向量乘计算任务中（特别地，当矩阵$\boldsymbol{A}$为宽矩阵时），与基于MDS码和LT码的CDC方案相比，基于低ϵ HBMC的CDC方案编译码时间更短。

4.2 任务时间

为分析分布式系统的计算时间，本文采用与文献[21,28]类似的数学模型。假定系统各节点之间计算相互独立，节点计算s次向量乘的计算时间为T_s，其概率分布$F_s(t)$服从指数分布，即

$$F_s(t)=\begin{cases}1-\mathrm{e}^{-\lambda\left(\frac{t}{s}-\alpha\right)}, & t\geqslant s\alpha\\ 0, & t<s\alpha\end{cases}\tag{1}$$

其中，λ和α是常数，λ与节点的计算性能相关，α表示节点计算一次向量乘所需的最小时间，故$T_s\geqslant s\alpha$。在F-HB中，节点最多需要进行w次向量乘计算。将F-HB完成分布式计算的时间记为T_{F}。显然，其概率分布可以表示为

$$\Pr\{T_{\mathrm{F}}\leqslant t\}=\begin{cases}\sum\limits_{d=\tilde{d}}^{k}F_w(t)^d\left(1-F_w(t)\right)^{k-d}\binom{k}{d}, & t\geqslant w\alpha\\ 0, & t<w\alpha\end{cases}\tag{2}$$

因此，F-HB完成分布式计算的期望时间为

$$E(T_{\mathrm{F}})=\int_0^{\infty}(1-\Pr\{T_{\mathrm{F}}\leqslant t\})\mathrm{d}t\tag{3}$$

在R-HB中，主节点会将单次向量乘任务交由当前空闲节点计算。节点计算单次向量乘的时间概率分布为$F_1(t)$。由于节点所计算的向量乘相互独立，此时$F_s(t)$服从s阶埃尔朗分布，即$F_s(t)$~Erlang(s,λ)。各节点最多计算$(1+\epsilon)m$次向量乘，且系统收到的返回结果不会超过$(1+\epsilon)m$个。将第i个节点在t时刻计算的向量乘次数记为$S_i(t)$，则

$$S_i(t)\backsim\begin{pmatrix}0 & 1 & \cdots\\ 1-F_1(t) & F_1(t)-F_2(t) & \cdots\\ (1+\epsilon)m-1 & (1+\epsilon)m & \\ F_{(1+\epsilon)m-1}(t)-F_{(1+\epsilon)m}(t) & F_{(1+\epsilon)m}(t) & \end{pmatrix}\tag{4}$$

将t时刻R-HB中前k'个节点计算的向量乘次数记为$M_{k'}(t)$，其中$M_{k'}(t)=\sum\limits_{i=1}^{k'}S_i(t)$，系统总共完成$M_k(t)$次计算。当系统的计算次数$M_k(t)\geqslant(1+\epsilon)m$时，可近似看作计算完成。用$T_{\mathrm{R}}$表示R-HB完成分布式计算的时间，其概率分布可以描述为

$$\Pr\{T_{\mathrm{R}}\leqslant t\}=\Pr\{M_k(t)\geqslant(1+\epsilon)m\}=1-\Pr\{M_k(t)<(1+\epsilon)m\}\tag{5}$$

本文使用递推算法求解式(5)，其递推关系可以用如图5所示的Trellis图表征。

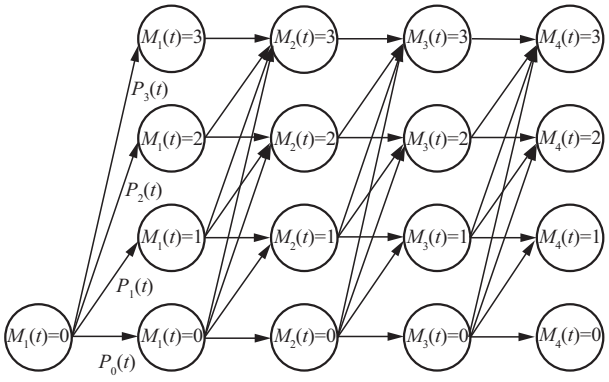


图5　R-HB 节点的 Trellis 图（$m=2,\epsilon=0.5,k=4$）

Trellis 图中的第 k' 列表示当前已知计算次数的节点总数为 k'，第 l 行表示已知节点共完成的计算次数 $M_k(t)=l$，其中 $k'\leqslant k$，$M_k(t)<(1+\epsilon)m$。Trellis 图中各状态发生概率由上一列状态概率转移得到。具体来说，t 时刻前 k' 个节点的总计算次数与前 $k'-1$ 个节点的总计算次数有关，即有

$$\Pr\left\{M_k(t)=l\right\}=\sum_{l'=0}^{l}\Pr\left\{M_{k'-1}(t)=l'\right\}\Pr\left\{S_i(t)=l-l'\right\} \tag{6}$$

若 $l=0$，则 $\Pr\left\{M_0(t)=l\right\}=1$，否则 $\Pr\{M_0(t)=l\}=0$。因此，可以通过

$$\Pr\left\{M_k(t)<(1+\epsilon)m\right\}=\sum_{l=0}^{(1+\epsilon)m-1}\Pr\left\{M_k(t)=l\right\} \tag{7}$$

对概率分布进行计算，同样，R-HB 完成分布式计算的期望时间为

$$E\left(T_{\mathrm{R}}\right)=\int_0^{\infty}(1-\Pr\left\{T_{\mathrm{R}}\leqslant t\right\})\mathrm{d}t \tag{8}$$

4.3　方案优化

在 F-HB 中，w 的取值与 c 的设置有关，会影响分布式计算的完成时间。假设系统中各节点在本地能够存储 z 个矩阵 $\boldsymbol{A}$，出现掉队的概率为 P_{stg}。若系统不存在掉队节点，应直接将 $(1+\epsilon)m$ 行数据分发给 k 个节点进行计算，此时 $c=1+\epsilon$。编码后数据总量不应超过节点存储空间之和，故 $cm\leqslant zmk$。综上可知

$$1+\epsilon\leqslant c\leqslant zk \tag{9}$$

在实际分布式系统中，计算任务与节点总数往往是固定的。在固定 k 与 m 的条件下，下调 c 会减少各节点的计算时间，但也会增大掉队节点的影响。理想情况下，期望冗余行数应恰好等于期望掉队节点分配的向量乘次数，即 $\left[c-(1+\epsilon)\right]m=wkP_{\mathrm{stg}}$，故此时 $c=\dfrac{1+\epsilon}{1-P_{\mathrm{stg}}}$ 为最优码率。

4.4　恢复阈值

在 F-HB 中，每个节点负责的编码子矩阵 $\boldsymbol{B}_i$ 的行数为 w，即每个节点完成计算后，向主节点返回 w 行计算结果。因此，d 个节点共计算 dw 行任务。

根据 HBMC 的译码特性，主节点至少收集 $(1+\epsilon)m$ 行计算结果，能够以高概率恢复最终结果 $\boldsymbol{y}=\boldsymbol{Ax}$。由此，需要满足 $dw\geqslant(1+\epsilon)m$。

因而，可解节点数 d 满足

$$d\geqslant\frac{(1+\epsilon)m}{w}$$

在 HBMC 的生成矩阵 $\boldsymbol{G}$ 中，任意 $(1+\epsilon)m$ 行线性无关[26]。在 F-HB 方案中，编码矩阵 $\boldsymbol{B}=\boldsymbol{GA}$，其行向量为 $\boldsymbol{G}$ 的行向量与 $\boldsymbol{A}$ 的列向量的线性组合。

由于 $\boldsymbol{A}$ 为原始信息矩阵，$\boldsymbol{G}$ 的行线性无关性可传递至 $\boldsymbol{B}$ 的行线性无关性。因此，当 $dw\geqslant(1+\epsilon)m$ 时，d 个节点的子矩阵 $\boldsymbol{B}_i$ 拼接后的矩阵（共 dw 行）中，必然存在至少 $(1+\epsilon)m$ 个线性无关的行，满足 HBMC 译码要求。

考虑到 HBMC 有概率译码失败，因此，F-HB 的恢复阈值 $\tilde{d}\approx\left\lceil\dfrac{(1+\epsilon)m}{w}\right\rceil$。相较于基于 MDS 码的经典 CDC 方案的恢复阈值 $\tilde{d}^*=\left\lceil\dfrac{m}{w}\right\rceil$，F-HB 的恢复阈值仅多出 $\epsilon\tilde{d}^*$ 个计算节点。

由于无速率特性，R-HB 不存在恢复阈值的理论界。当主节点至少收集 $(1+\epsilon)m$ 行计算结果后，能够以高概率恢复最终结果 $\boldsymbol{y}=\boldsymbol{Ax}$，而这些结果可能由多个甚至全体节点参与计算。

5　实验分析

本节展示 HBMC 译码成功概率的渐近下界和仿真结果，并通过数值计算与实验仿真证明所提方案的有效性。

5.1　译码成功概率

图 6 刻画了文献[26]中 HBMC 译码成功概率的渐近下界 $1-\mathrm{e}^{-\Omega\left(\epsilon^2 m\right)}$ 及 HBMC 在删除信道仿真中的译码成功概率。从图 6 可知，译码成功概率的实际仿真结果不低于渐近下界，符合预期，且随着 ϵ 的增大，二者趋于 1。随着 m 增大，在相同 ϵ 条件下，

HBMC 的译码成功概率有所提高。在基于 HBMC 的 CDC 方案中，掉队节点负责计算的数据可以看作删除信道中被删除的信息，而这些信息可以通过 HBMC 恢复。因此，F-HB 与 R-HB 的译码成功概率和 HBMC 保持一致，不低于渐近下界。

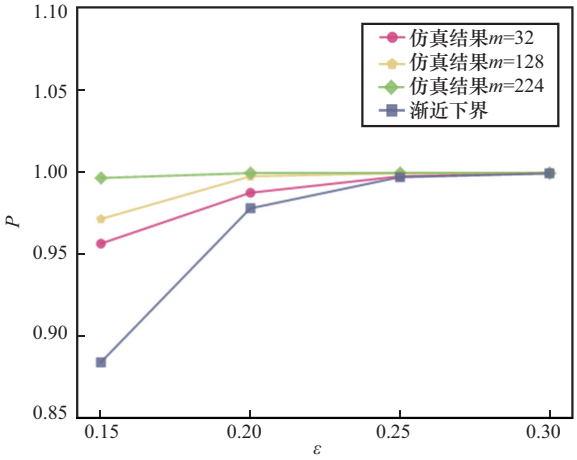


图 6　HBMC 译码成功概率

5.2　期望任务时间

本文通过数值计算，验证式(3)和式(8)的正确性。采用式(1)定义的指数分布模型完成计算时间的统计，并对比不同参数下 F-HB、R-HB 以及未编码分布式计算（uncoded distributed computing，UDC）方案的计算结果。

从图 7(a)可知，随着系统中节点数量增加，F-HB 与 R-HB 的期望计算时间均明显下降，而 UDC 案的期望任务时间下降速度缓慢或几乎不变。这表明，在 UDC 系统中，掉队节点造成的负面影响较大，降低了节点增加带来的性能增益。

λ 与节点的计算性能相关，λ 越大，表示系统中节点完成计算任务的速度越快，出现掉队现象的概率越低。从图 7 (b)可知，当 λ 较小时，F-HB 的期望任务时间明显低于 R-HB；当 λ 较大时，F-HB 的期望任务时间则高于 R-HB。这表明，在计算环境良好的系统中，R-HB 能够更有效地利用掉队节点的计算性能，减少计算资源浪费，实现更短的任务时间。

从图 7(c)可知，CDC 的期望计算时间会随计算任务规模的增大而增大。在 ϵ 固定的条件下，随着 m 增大，F-HB 与 R-HB 所需的额外计算次数增加，故期望计算时间增长幅度更大。

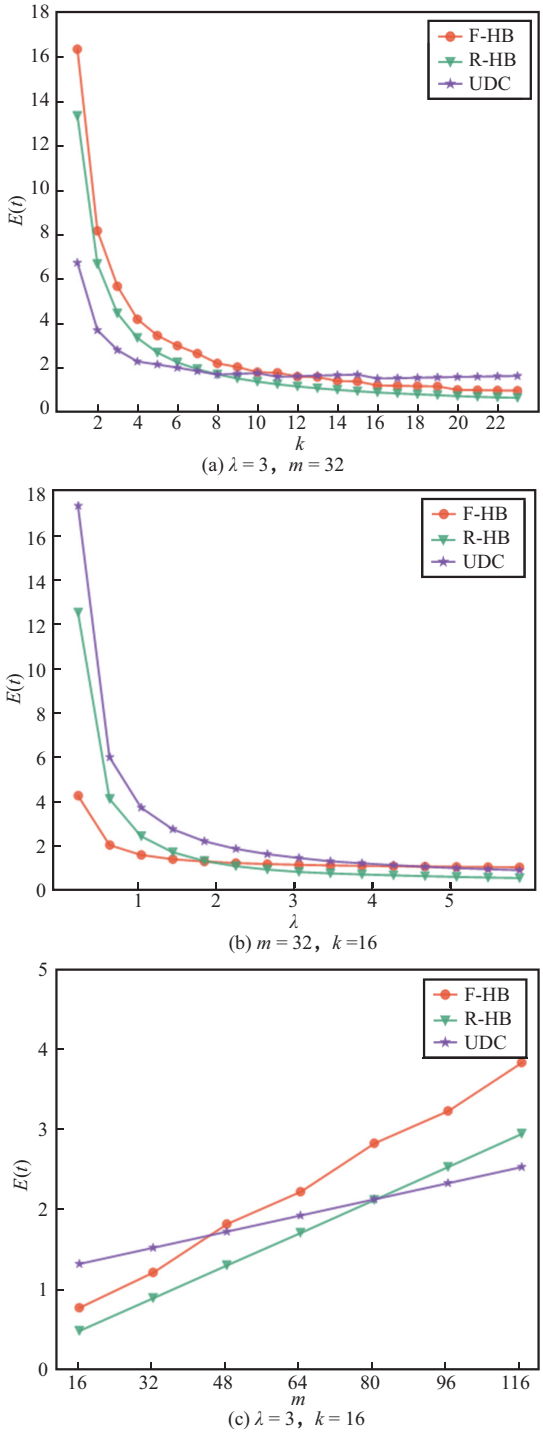


图 7　期望任务时间对比（$\epsilon=0.25$，$c=2.5$，$\alpha=0.2$）

5.3　仿真结果

实验采用 Intel(R) Xeon(R) Gold 6230 CPU @ 2.10 GHz 处理器、128 GB DDR4 内存的计算平台，在 Python3.11 的环境下进行仿真实验。

本文将所提 F-HB、R-HB 两类 CDC 方案部署在上述服务器上，并与基于固定码率 MDS 码的

CDC 方案[11]（简称 F-MDS）、基于固定码率低权重码的 CDC 方案[14]（简称 F-LW）、基于多项式码的 CDC 方案[22]（简称 RSP）、基于无速率 LT 码的 CDC 方案[23]（简称 R-LT）、基于固定码率 SC-LDGM 码的 CDC 方案[29]（简称 F-LDGM）进行性能对比。各对比方案实验设置如表 2 所示。

表 2　　各对比方案实验设置

对比方案	编码方式	关键参数
UDC[1]	不编码	无相应编码参数
F-MDS[11]	RS 码	$c=\dfrac{1}{1-P_{\mathrm{stg}}}$
F-LW[14]	LW 码	$c=\dfrac{k}{k_A}\approx\dfrac{1}{1-P_{\mathrm{stg}}}$
RSP[22]	拉格朗日多项式	$c=\dfrac{k\left\lceil\dfrac{m}{k(1-P_{\mathrm{stg}})}\right\rceil}{m}\approx\dfrac{1}{1-P_{\mathrm{stg}}}$
R-LT[23]	LT 码	$\epsilon=\dfrac{\ln\left(\dfrac{m}{\delta}\right)^2}{\sqrt{m}}$
F-HB[25]	HBMC	$c=\dfrac{1+\epsilon}{1-P_{\mathrm{stg}}}$，$\epsilon=\arg\min\left\{\epsilon\left\vert 1-\mathrm{e}^{-\Omega\left(\epsilon^2 m\right)}\geqslant 0.95\right.\right\}$
R-HB[25]	HBMC	$\epsilon=\arg\min\left\{\epsilon\left\vert 1-\mathrm{e}^{-\Omega\left(\epsilon^2 m\right)}\geqslant 0.95\right.\right\}$
F-LDGM[29]	SC-LDGM	$c=\dfrac{1}{1-P_{\mathrm{stg}}}$，$\rho=0.012$

注：方案中未提及的参数，可由上述关键参数及给定参数推导计算得出。

实验通过 k 个节点计算矩阵−向量乘 $\boldsymbol{Ax}$。为模拟实际分布式系统中的掉队现象，本文主要参考文献[30]的环境设置进行实验。当节点发生掉队问题时，其计算速度远低于其他节点，或因争夺资源、网络拥塞等原因，导致其结果返回时间增大，甚至是无限大[7]。参考 AWS EC2 实际的计算延迟特性[31]，实验采用尺度参数 $\theta=0.2$、形状参数 $\kappa=6$ 的伽马（Gamma）分布模拟计算延迟。

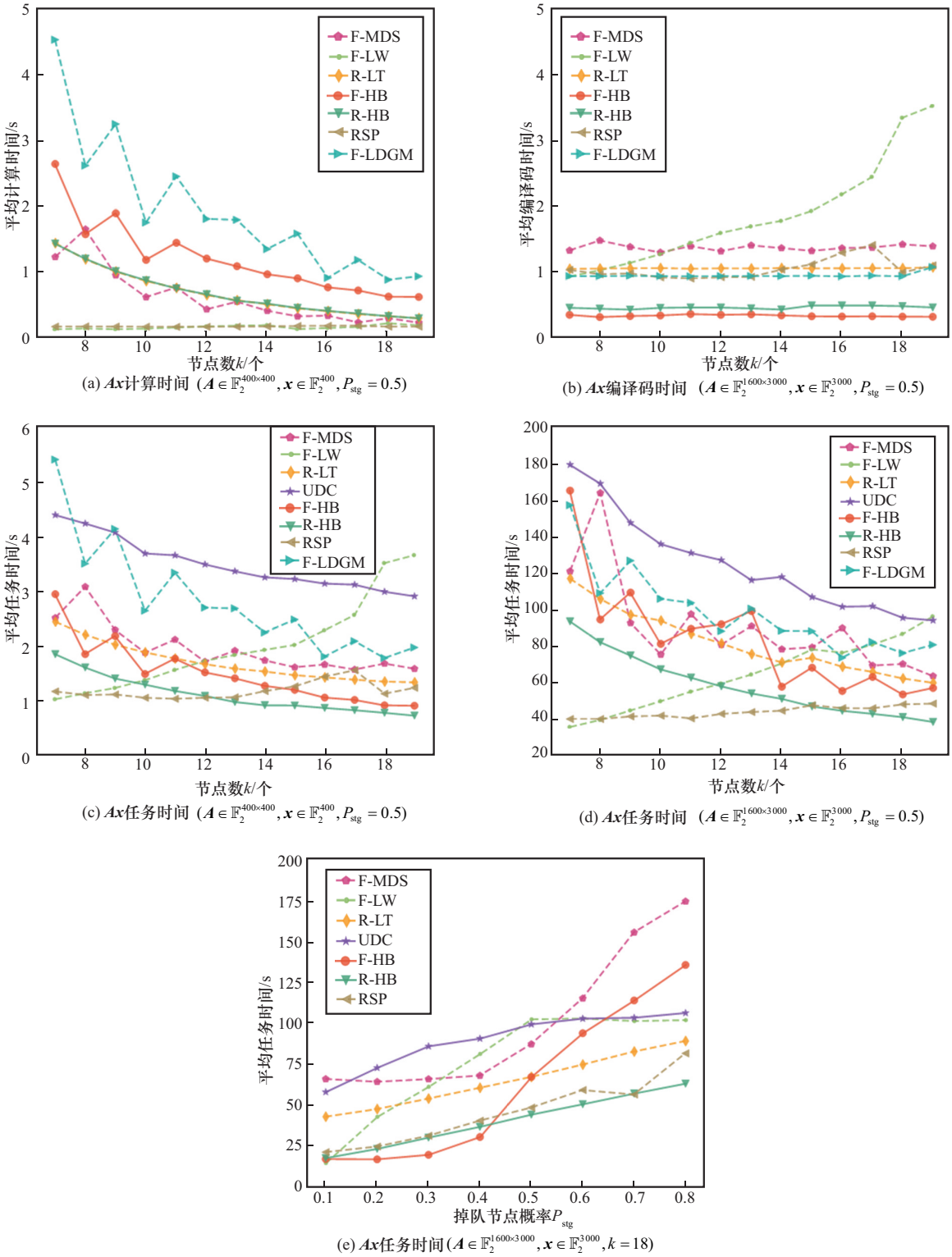
根据 CDC 方案的不同，实验将计算矩阵 $\boldsymbol{A}$ 进行编码，并分配给各节点进行计算。默认节点已知计算向量 $\boldsymbol{x}$。各个节点会在上述服务器上进行串行计算。本文通过记录各个节点的索引及计算时间，模拟不同 CDC 方案的并行计算过程。在仿真环境中，相比编译码时间和计算时间，通信时延较低，故本文实验不作通信时延的讨论。不同方案、参数下各进行 1 000 次仿真实验，并取平均完成一次计算任务的时间作为衡量 CDC 方案计算性能的度量。

图 8(a)描绘了各 CDC 方案的平均计算时间与节点数 k 的关系。理论上，参与计算的节点数越多，每个计算节点所负责的计算任务越少，计算时间应越短。然而，在 F-LW、RSP 中，节点的任务大小还取决于当前系统掉队节点的数量。在固定 P_{stg} 的情况下，随着参与节点数的增加，上述系统中每个节点所分配的计算任务并未减少，计算时间也未降低，系统中节点存在掉队问题，使各 CDC 方案的计算时间曲线呈现出波动的趋势。固定码率 CDC 方案的计算时间，由接收到的 d 个结果中的最慢结果返回时间决定。当非掉队节点个数小于 $\tilde{d}$ 时，系统计算时间会显著上升。故该类方案的曲线的波动幅度大。与 F-HB 相比，F-MDS 的恢复阈值更低，因此，其计算时间会更短。F-LW 通过信息叠加而非编码冗余的方法处理掉队问题，其节点的计算任务较少，计算时间较短。RSP 的恢复阈值达到理论最优恢复阈值，略优于所提方案，计算时间也较短。由于无速率 CDC 方案的任务颗粒度细，系统中的掉队节点能够参与部分计算，其计算时间较 F-HB 更短，接近 F-MDS。

基于低 ϵ 的 HBMC 编译码复杂度可以低于其他编码方法。从图 8(b)可以看出，对于实际的矩阵−向量乘任务 $\boldsymbol{Ax}$，基于 HBMC 的 CDC 方案平均编译码时间更短。随着掉队节点数增加，F-LW 需要叠加的信息增多，故编译码时间也显著增加。

从图 8(c)和图 8(d)可以看出，由于掉队节点的影响，在不同任务规模下，CDC 方案的计算性能均优于 UDC 方案。基于 HBMC 的 CDC 方案平均编译码时间更短，因此 F-HB、R-HB 的计算性能整体上优于 F-MDS、F-LDGM 和 R-LT。在固定 P_{stg} 的条件下，随着计算节点个数的增加，基于 HBMC 的 CDC 方案并不会增加编译码时间，且会显著减少计算时间。在计算节点数较多的环境下，其计算性能也优于 F-LW、RSP。

从图 8(e)可以看出，所提方案在不同 P_{stg} 的计算环境下，均有较优的性能表现，其鲁棒性较好。相较 UDC 方案，F-HB 与 R-HB 的任务时间分别降低 68% 与 74%。



(a) $\boldsymbol{Ax}$计算时间 $(\boldsymbol{A}\in\mathbb{F}_2^{400\times400},\boldsymbol{x}\in\mathbb{F}_2^{400},P_{stg}=0.5)$

(b) $\boldsymbol{Ax}$编译码时间 $(\boldsymbol{A}\in\mathbb{F}_2^{1600\times3000},\boldsymbol{x}\in\mathbb{F}_2^{3000},P_{stg}=0.5)$

(c) $\boldsymbol{Ax}$任务时间 $(\boldsymbol{A}\in\mathbb{F}_2^{400\times400},\boldsymbol{x}\in\mathbb{F}_2^{400},P_{stg}=0.5)$

(d) $\boldsymbol{Ax}$任务时间 $(\boldsymbol{A}\in\mathbb{F}_2^{1600\times3000},\boldsymbol{x}\in\mathbb{F}_2^{3000},P_{stg}=0.5)$

(e) $\boldsymbol{Ax}$任务时间$(\boldsymbol{A}\in\mathbb{F}_2^{1600\times3000},\boldsymbol{x}\in\mathbb{F}_2^{3000},k=18)$

图8　仿真时间对比

6　结束语

针对分布式计算中的掉队节点问题，基于HBMC，本文提出F-HB以及R-HB两种加速矩阵–向量乘计算的CDC方案，并从理论分析和仿真实验两个方面证明所提方案的有效性。相较UDC方案，F-HB与R-HB的任务时间分别降低68%与74%。所提两类方案均可推广到诸如傅里叶变换、梯度下降等非线性的分布式计算系统中。

参考文献：

[1] Cristea V, Dobre C, Stratan C, et al. Large-Scale Distributed Computing and Applications[M]. Hershey: IGI Global, 2010.

[2] 徐明, 张保俊, 伍益明, 等. 面向网络攻击和隐私保护的多智能体系统分布式共识算法[J]. 通信学报, 2023, 44(3): 117-127.
Xu M, Zhang B J, Wu Y M, et al. Cyber attacks and privacy protection distributed consensus algorithm for multi-agent systems[J]. Journal on Communications, 2023, 44(3): 117-127.

[3] 张晓旭, 陈宇辰, 哈冠雄, 等. 基于分布式存储的外包 EHR 隐私保护分类审计方案[J]. 通信学报, 2024, 45(9): 26-39.
Zhang X X, Chen Y C, Ha G X, et al. Classification auditting scheme for privacy protection of outsourced EHR based on distributed storage[J]. Journal on Communications, 2024, 45(9): 26-39.

[4] 柴蓉, 艾莉萍, 杨泞渝, 等. 支持拜占庭容错的分布式物联网访问控制机制[J]. 通信学报, 2025, 46(4): 174-186.
Chai R, Ai L P, Yang N Y, et al. Byzantine fault-tolerant distributed access control mechanism for the Internet of Things[J]. Journal on Communications, 2025, 46(4): 174-186.

[5] Cartor R, D'Oliveira R G L, Rouayheb S E, et al. Secure distributed matrix multiplication with precomputation[C]//Proceedings of the 2024 IEEE International Symposium on Information Theory (ISIT). Piscataway: IEEE Press, 2024: 2568-2573.

[6] Malak D. Distributed structured matrix multiplication[C]//Proceedings of the 2024 IEEE International Symposium on Information Theory (ISIT). Piscataway: IEEE Press, 2024: 2550-2555.

[7] Dean J, Barroso L A. The tail at scale[J]. Communications of the ACM, 2013, 56(2): 74-80.

[8] Blumofe R D, Leiserson C E. Scheduling multithreaded computations by work stealing[J]. Journal of the ACM, 1999, 46(5): 720-748.

[9] Attia M A, Tandon R. Combating computational heterogeneity in large-scale distributed computing via work exchange[J].arXiv Preprint, arXiv: 1711.08452, 2017.

[10] Wang D, Joshi G, Wornell G. Using straggler replication to reduce latency in large-scale parallel computing[J]. ACM SIGMETRICS Performance Evaluation Review, 2015, 43(3): 7-11.

[11] Lee K, Lam M, Pedarsani R, et al. Speeding up distributed machine learning using codes[J]. IEEE Transactions on Information Theory, 2018, 64(3): 1514-1529.

[12] Ferdinand N S, Draper S C. Anytime coding for distributed computation[C]//Proceedings of the 2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton). Piscataway: IEEE Press, 2016: 954-960.

[13] Wang S, Liu J, Shroff N. Coded sparse matrix multiplication[C]//Proceedings of the 35th International Conference on Machine Learning. New York: PMLR, 2018: 5152-5160.

[14] Das A B, Ramamoorthy A, Love D J, et al. Distributed matrix computations with low-weight encodings[J]. IEEE Journal on Selected Areas in Information Theory, 2023, 4: 363-378.

[15] Severinson A, Graell i Amat A, Rosnes E. Block-diagonal and LT codes for distributed computing with straggling servers[J]. IEEE Transactions on Communications, 2019, 67(3): 1739-1753.

[16] Yang X, Jiang M, Zhao C M. LT codes with feedback: accelerate the distributed matrix-vector multiplication with stragglers[C]//Proceedings of the 2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC). Piscataway: IEEE Press, 2019: 1-6.

[17] Choi J. Wireless distributed matrix-vector multiplication using over-the-air computation and analog coding[J]. IEEE Transactions on Wireless Communications, 2024, 23(8): 9826-9838.

[18] Fidalgo-Díaz A, Martínez-Peñas U. Distributed matrix multiplication with straggler tolerance using algebraic function fields[J]. IEEE Transactions on Information Theory, 2025, 71(2): 996-1006.

[19] Jahani-Nezhad T, Ali Maddah-Ali M. Berrut approximated coded computing: straggler resistance beyond polynomial computing[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2023, 45(1): 111-122.

[20] Jahani-Nezhad T, Ali Maddah-Ali M. CodedSketch: a coding scheme for distributed computation of approximated matrix multiplication[J]. IEEE Transactions on Information Theory, 2021, 67(6): 4185-4196.

[21] Ferdinand N, Draper S C. Hierarchical coded computation[C]//Proceedings of the 2018 IEEE International Symposium on Information Theory (ISIT). Piscataway: IEEE Press, 2018: 1620-1624.

[22] Qiu H M, Zhu K, Luong N C, et al. Coded distributed computing for resilient, secure and private matrix-vector multiplication in edge-enabled metaverse[J]. IEEE Transactions on Cognitive Communications and Networking, 2024, 10(5): 1944-1958.

[23] Mallick A, Chaudhari M, Sheth U, et al. Rateless codes for near-perfect load balancing in distributed matrix-vector multiplication[C]// Proceedings of the Abstracts of the 2020 SIGMETRICS/Performance Joint International Conference on Measurement and Modeling of Computer Systems. New York: ACM Press, 2020: 95-96.

[24] Severinson A, i Amat A G, Rosnes E, et al. A droplet approach based on raptor codes for distributed computing with straggling servers[C]// Proceedings of the 2018 IEEE 10th International Symposium on Turbo Codes & Iterative Information Processing (ISTC). Piscataway: IEEE Press, 2018: 1-5.

[25] Alon N, Edmonds J, Luby M. Linear time erasure codes with nearly

optimal recovery[C]//Proceedings of IEEE 36th Annual Foundations of Computer Science. Piscataway: IEEE Press, 1995: 512-519.

[26] Edmonds J, Luby M. Erasure codes with a hierarchical bundle structure[J]. IEEE Transactions on Information Theory, 2017. DOI: 10.1109/TIT.2017.2777836.

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

[28] Ozfatura E, Gunduz D, Ulukus S. Speeding up distributed gradient descent by utilizing non-persistent stragglers[C]//Proceedings of the 2019 IEEE International Symposium on Information Theory (ISIT). Piscataway: IEEE Press, 2019: 2729-2733.

[29] Cai S H, Lin W C, Yao X, et al. Systematic convolutional low density generator matrix code[J]. IEEE Transactions on Information Theory, 2021, 67(6): 3752-3764.

[30] Kiani S, Adikari T, Draper S C. Hierarchical coded elastic computing[C]//Proceedings of the ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Piscataway: IEEE Press, 2021: 4045-4049.

[31] Lê-quôc A, Fiedler M, Cabanilla C. The top 5 AWS EC2 performance problems[R]. 2013.

[作者简介]

韦宝典（1976– ），男，广西贵港人，博士，中山大学副教授、硕士生导师，主要研究方向为信息论与信道编码、信息安全等。

莫肇豪（2000– ），男，广东阳江人，中山大学硕士生，主要研究方向为信道编码、语义通信。

马啸（1968– ），男，河南焦作人，博士，中山大学教授、博士生导师，主要研究方向为信息论与信道编码、编码调制技术、无线通信和光通信等。