基于组合设计及其 q-类似物的缓存辅助通信方案

Shailja Agrawal，K V Sushena Sree，Prasad Krishnan，Abhinav Vaishya，Srikar Kale

海得拉巴国际信息技术学院

电子邮件：{shailja.agrawal@research.，sushena.sree@research.，prasad.krishnan@ ,

abhinav.vaishya@research. , srikar.kale@research. }iiit.ac.in

arXiv:2302.03452v1 [cs.IT] 7 Feb 2023

摘要

我们考虑标准的广播设置：单个服务器向多个客户端广播信息，每个客户端都具有一定容量的本地存储（称为缓存），可用于存储服务器上可用文件的部分内容。集中式编码缓存框架由缓存阶段和交付阶段组成，这两个阶段经过精心设计，以便最优地联合利用缓存和信道。在以往的文献中，人们使用了各种组合结构来构造编码缓存方案。许多现有构造的主要缺点之一是子包化级别较高；它表示为使方案获得编码增益，文件需要被分割的次数。在这项工作中，我们利用一种新的二进制矩阵模型，基于多种类型的组合设计及其 q-类比（也称为子空间设计），提出了若干新颖的编码缓存构造。尽管本工作中构造的大多数方案（基于已有设计）对缓存的需求较高，但其速率要么为常数，要么随参数增大而降低；此外，这些方案所需的子包化级别具有竞争力且较低，而这对于编码缓存的实际应用而言是极其重要的特性。我们还将这些构造应用于 MapReduce 分布式计算框架，该框架包含三个阶段：映射阶段、洗牌阶段和归约阶段。利用我们的二进制矩阵框架，我们提出了一种新的、简单且通用的编码数据洗牌方案。将基于设计的构造与这一新洗牌方案结合使用，我们获得了新的编码计算方案；这些方案具有较低的文件复杂度，对于相同参数，其通信负载仅略高于最优方案。我们还表明，这些方案可以自然地扩展到存在完全掉队节点和部分掉队节点的情形。

I. 引言

客户端存在侧信息的多终端广播通信，是许多现代通信问题的典型场景，包括卫星通信、内容分发网络和分布式计算。对于每个客户端都具有本地存储（称为缓存）的广播场景， [3]提出了一种联合缓存与交付设计，称为编码缓存。 [3], 中的编码缓存场景包含 K 个客户端，每个客户端都拥有一定的本地存储（即其缓存），并通过一条无差错广播链路连接到一台服务器。服务器包含一个由 N 个等大小文件组成的文件库，而每个客户端缓存可以

本工作的部分内容曾发表于2019年IEEE信息论国际研讨会（ISIT），法国巴黎 [1]以及2020年IEEE信息论国际研讨会（ISIT），美国加利福尼亚州洛杉矶， [2]的会议论文集。

存储至多 M 个文件（其中 $M \leq N$满足某个条件）。 [3] 中的编码缓存框架分为两个阶段：放置阶段（即填充缓存的阶段）和交付阶段。在交付阶段（高峰时段），用户会提出需求。在 [3], 的编码缓存范式中，服务器广播编码传输，使所有用户的需求都得到满足。与 [3],一样，编码缓存方案的速率 R 定义为服务器传输的比特数量与每个文件大小之比。设计一个优秀的编码缓存方案，需要联合设计缓存／预取阶段，以及一种能够有效利用客户端缓存子文件所带来的编码机会的传输方案，从而使方案的速率最小化。

对于这一编码缓存系统， [3] 的作者提出了一种编码缓存方案，其中包括对文件的子文件进行精心放置，并设计了恰当的传输方案。研究表明， [3]方案的速率为 $R = {}^{K}(1-\frac{M}{N})\,1+{}^{MK}_{N}$，相较于未编码传输速率(, 其增益为 $1 + {}^{MK}_{N} = \Theta(K)$；其中 $\frac{M}{N}$) 为常数。研究还表明，在 [4], 中，假设采用未编码缓存放置且满足 $N \geq K$ 时，Ali-Niesen 方案 [3] 对于给定的缓存大小 M 是最优的。

有趣的是， [3]提出的范式已扩展到多种涉及多个终端的缓存辅助通信场景，例如设备到设备（D2D）通信网络 [5],、分布式计算[6], 以及无线干扰信道中的干扰管理 [7]。编码缓存范式发挥重要作用的一个场景，是广受欢迎的 MapReduce 分布式计算框架[6]。在 MapReduce 框架中，大型数据文件被划分为较小的数据分片，再将这些分片分配给不同的服务器，以分布式方式进行处理。MapReduce 包含两个主要阶段：映射和归约，两者之间由第三个数据洗牌阶段衔接。在映射阶段，每个数据分片由一个或多个服务器处理，通过映射函数生成一些中间值（IVA）。下一步，服务器交换这些中间值（称为数据洗牌），以便利用归约函数在各服务器间以分布式方式计算最终输出。正如 [8] 和 [9], 所指出的，数据洗牌是决定原始 MapReduce 框架性能的重要阶段；在该框架的洗牌阶段，中间值以未编码的方式传输。 [10], 表明，可以利用这样一个事实： r 个经过精心挑选的不同节点映射相同的子文件（因而具有相同的中间值），在洗牌过程之前将这些中间值编码在一起，从而大幅降低通信负载。参数 r 称为计算负载，它表示降低通信负载所需付出的代价。这种采用编码洗牌阶段的新框架称为编码 MapReduce。 [11],进一步研究了这一模型，并证明 [11]提出的编码 MapReduce 方案所达到的通信负载是最优的。在 [12]–[14], 中， [10], [11] 提出的模型得到进一步扩展，以研究这样的编码 MapReduce 方案：节点无需计算所有已存储子文件的中间值，即可完成归约任务。 [13], [14], 推导了存储、计算和通信负载之间的权衡，并提出了一个达到这一权衡的最优方案。 [15], [16] 研究了计算过程中存在掉队者时的编码分布式计算方案，针对的是计算线性函数的情形。掉队者是指速度缓慢或完全无法完成映射任务的节点。随后， [17], [18], 将 [11]的编码 MapReduce 模型扩展到存在完全掉队节点和部分掉队节点时的任意函数计算。

尽管编码缓存中的 Ali-Niesen 方案 [3] 和编码 MapReduce 方案 [11] 已被证明是最优的，但所需的子分组化级别——即为实现编码传输而必须将每个文件拆分成的部分数量——会随 K 呈指数增长（缓存比例固定时）。在 [3], 的编码缓存方案设计中，每个文件都必须在该方案中拆分为 $\binom{K}{MK/N}$ 个子文件；当客户端数量 K （ M/N 固定时）随着 K 增大，这一数量可能呈指数增长。编码分布式计算场景中也存在类似问题 [11]。这对方案的实际实现造成了重大障碍（关于这些问题的讨论，参见 [19] 的第一节）。为解决这一问题，人们提出了许多研究方案，采用了多种不同方法，包括基于用户分组的方法 [20],、二分图的强边着色 [21], 、一种称为放置-传送阵列（PDA）的新型组合结构 [22] 以及其他改进构造 [23], [24], 、基于超图的方案 [25], 、图的诱导匹配 [26], 、可分解设计 [27], 、组合设计 [28], 、正交数组 [29], [30],、基于射影几何的构造 [19],[31], , 甚至是不需要文件拆分的构造 [32]。这份清单或许并不完整，而且仍在不断扩充。尽管这方面的大多数文献讨论的是编码缓存范式，但其中许多成果都可通过简单技术推广到 MapReduce 场景 [14], [33], [34]。近期也有人提出了通过巧妙地“提升”现有 PDA、以此从现有 PDA 构造新 PDA 的方法 [35], [36]。无论如何，任何编码缓存方案的三个核心性能参数——缓存比例 M/N、速率 R 和子包化级别 F—— 之间确实存在权衡，这一点已由 [4], [19], [37] 中编码缓存方案速率的下界所揭示。 [37] 中的下界适用于基于 PDA 的方案，而 [4], [19] 中的界限则属于信息论性质，适用于所有编码缓存方案（包括非线性方案）。具体而言，这些下界表明，对于合理数量的客户端，这三个参数中至多只能同时降低两个。

上述文献中的许多编码缓存方案都致力于构造低速率、低缓存比例的方案，但这会导致较高的子包化级别（通常随用户数量 K 呈指数增长）。这似乎是很自然的选择，因为在实践中，我们预计本地缓存的容量远小于文件库本身，而文件库可能极其庞大（例如，在内容分发网络中，为用户提供大量可供下载的内容选择）。不过，在某些网络场景中，网络带宽可能是首要考虑因素，而本地存储则可能足以缓存完整文件库中更大的比例。例如，某个网络为拥有充足本地存储空间的小规模受众提供热门视频；又如，某个网络的带宽成本高得难以承受或十分稀缺，而缓存本身的成本却可能很低，例如关键环境中的通信。文献中已有一些编码方案适用于灵活的存储容量，因此也适用于较高的缓存比例。本文提出若干新的编码缓存方案，适用于缓存比例高、子包化级别低且速率低的情形。我们还给出渐近比较（当 K 增大时）和数值比较，将这些方案与一些重要的现有方案及已有下界进行比较，展示了我们的方案在多种情况下的优势。

我们使用编码缓存的二元矩阵模型来表述构造方法。在第二节中，我们介绍如何使用恒定行重量二元矩阵来描述编码缓存方案的概念。我们将这些矩阵称为缓存矩阵。二进制矩阵中的“1”表示用户尚未缓存的子文件。缓存矩阵中的单位子矩阵

对应于这样的传输：它使参与该传输的客户端能够从中准确解码出各自缺失的一个子文件。因此，使用单位子矩阵“覆盖”缓存矩阵中的“1”，即可得到有效的传输方案。我们提出的基于二进制矩阵的框架，本质上等同于 [22] 中提出的 PDA框架。不过，这种视角的优势在于，它将搜索优良缓存方案的空间大幅拓展到了所有恒定行重量二元矩阵的集合。

接下来，我们采用二元矩阵模型，利用各种组合设计及其 q-类比构造新颖的缓存方案，这类方案也称为子空间设计。为此，第三节介绍与组合设计及其 q-类比相关的重要术语。在第五至第八节中，我们详细介绍如何利用不同的组合设计和子空间设计来构造缓存矩阵。具体而言，我们采用Steiner系统（具有特殊性质的t-设计）、平衡不完全区组设计、横截设计以及Steiner系统的 q-类比来构造缓存矩阵。当我们在这些构造中采用《组合学》文献中的现有设计时，所得缓存方案要求未缓存比例较低，即$1-\frac{M}{N}=\Theta(\frac{1}{K^i})$ $i=\frac{1}{2}$，1。这是一项缺点。不过，作为交换，这些方案的速率和子分包级别大幅降低，速率可达到常数甚至更低，而子分包级别至多仅随 K线性增长。第四节总结了所有构造，并在适用的情况下，讨论了它们在客户端数量不断增加时的渐近性质。

— — –

我们进一步表明，所设计的二进制矩阵也可以用作 MapReduce 框架的方案。在第九节简要回顾编码MapReduce设置后，我们在第十节介绍分布式计算中的计算矩阵概念。针对这一设置，我们在第 X-A 节提出一种新的简单传输方案。我们将 [11] 中展示的最优编码 MapReduce 方案解释为一种基于二进制矩阵的方案，并表明，对于该方案，我们的数据洗牌方案所实现的负载严格小于最优负载的两倍。然而，我们的方案在数据洗牌阶段的复杂度较低，因为它避免了将中间值进一步拆分成更小的数据包。我们在第 X-A1 节讨论这些内容。我们表明，第五至第八节所介绍的二进制矩阵构造能够得到编码MapReduce方案，并给出其参数。与最优方案相比，这些方案的通信负载较高，但文件复杂度非常低（第 X-C 节）。默认情况下，我们的新方案不保证通信负载均衡，也就是说，数据洗牌阶段并非所有服务器都会参与传输；不过，可以通过在适当定义的图上寻找完美匹配来实现这种负载均衡（第 X-B 节）。有趣的是，我们可以利用原始方案中负载不均衡这一特点来抵御掉队者。在第十一节，我们将方案扩展到存在掉队者的分布式计算场景。在完全掉队场景中，我们假设某些节点（数量不超过由计算矩阵性质决定的阈值）不可用或发生故障。对于我们在第十一节 A 中考虑的这一完全掉队模型，我们给出一种修订方案，其通信负载增加的倍数仅取决于每个节点所归约函数数量的增加。对于数量固定的部分掉队者（并非完全不可用、只是运行缓慢的节点），我们在第十一节 B 中表明，只需对方案稍作修改即可使其正常运行，且不会增加通信负载。

在第 XIII 节中，我们给出数值结果，将用于编码缓存和分布式计算的基于设计的方案，与这两种框架中的一些基准方案进行比较。对于编码缓存，我们使

我们的方案与这些基准方案在客户端数量和缓存比例（介于 0.75 到 0.9 之间）方面尽可能接近，并比较子分组化和速率。结果见表 IV 和表 V。对于 MapReduce，我们比较了本文提出的新方案与一些基准方案在无掉队者和掉队者场景下的表现。结果见表 VI 和表 VII。总体而言，我们的方案在子包化级别方面具有优势，但代价是通信负载略有增加。我们在第十二节总结了本文的贡献，并提出了一些有前景的未来研究方向。

符号与术语：对于任意正整数 N，我们用 $[N]$ 表示集合 $\{1,\ldots,N\}$。对于集合 $\mathcal{X}$和某个正整数 $t \leq |\mathcal{X}|$，我们用 $\binom{\mathcal{X}}{t}$表示 $\mathcal{X}$的所有大小为 t的子集。二项式系数记作 $\binom{n}{r} \triangleq {}_{r}{}^{n}!\,!(n-r)!$，其中 $n \geq r \geq 0$。对于一个矩阵 A，其行由有限集合 $\mathcal{R}$ 索引，列由有限集合 $\mathcal{C}$索引，第 r^{th} 行（$r \in \mathcal{R}$）和第 l^{th} 列（$l \in \mathcal{C}$）上的元素记作 $A(r,l)$。对于集合 A、 B， $A\backslash B$ 表示属于 A 但不属于 B 的元素。对于某个元素 i，我们也将 $A\backslash\{i\}$ 记作 $A\backslash i$。对于$j \in \{0,1,\ldots,(k-1)\}$，我们将$(j+1) mod\ k$记作$(j \oplus_k 1)$。对于非负整数 $k \leq v$和素数幂q，记号 $\left\langle v,k \right\rangle$表示高斯二项式系数，其定义为 $\left\langle v,k \right\rangle \triangleq (q^v-1)...(q^{v-k+1}-1)\ (q^k-1)...(q-1)\cdot$。为方便起见，我们在此记号中省略 q。这个数 $\left\langle v,k \right\rangle$ 也等于任意一个 v 维向量空间中维数为 k 的子空间数量，其中该向量空间定义在有限域 $\mathbb{F}_q$上，而该有限域有 q 个元素。

II. 二进制矩阵与编码缓存

本节介绍二进制矩阵的单位子矩阵覆盖这一概念，并说明具有恒定行重且存在单位子矩阵覆盖的二进制矩阵可以构成一种编码缓存方案。

为使表述更加严谨，我们先定义一些简单的量，并为它们赋予相应术语。我们将元素取自 $\{0,1\}$ 的矩阵称为二进制矩阵。形式上，矩阵 A 的子矩阵可由行索引的一个子集和列索引的一个子集指定，并分别用它们为 B的行和列编索引。若矩阵 A 的一个 $l \times l$ 子矩阵 B 的列对应于经过某种排列的 l 阶单位矩阵，则称 A 是一个大小为 l的单位子矩阵 。对于二进制矩阵 A，若行 i、列 j 上的非零元素 $A(i,\ j) = 1$ 所对应的行索引和列索引分别是单位子矩阵 B 中的某个行索引和列索引 i 和 j，则称该元素被该单位子矩阵覆盖 。若矩阵 A 中的某个非零元素同时被两个不同的单位子矩阵 B_1 和 B_2 覆盖，则称这两个子矩阵重叠。

定义 1（不重叠单位子矩阵覆盖）。 考虑矩阵 A 的一个由 S 个单位子矩阵组成的集合 $\mathfrak{C} = \{C_1,...,C_S\}$，其中 A 中的每个非零元素至少被一个 $C_i \in \mathfrak{C}$ 覆盖。则称 $\mathfrak{C}$ 为 C 的单位子矩阵覆盖。此外，如果 $\mathfrak{C}$ 中任意两个单位子矩阵均不重叠，则称 $\mathfrak{C}$ 为不重叠单位子矩阵覆盖。

现在回顾 [3]中的编码缓存系统模型。在 [3], 的编码缓存系统中，有一组由某个集合 $\mathcal{U}$ 索引的 K 个客户端，该集合的大小为 K，此外还有一台服务器。服务器上有一个由文件组成的文件库，其中包含 N 个大小相同的文件，记为 $W_i:\ \forall i \in [N]$。每个客户端都拥有本地存储（称为缓存），可存储文件库中比例为 M/N 的内容（称为缓存比例），其中 $M \in \{1,\ldots,N\}$。

客户端通过无差错共享链路连接到服务器。每个文件划分为 F 个大小相同且互不重叠的子文件，其中 F 称为子包化级别。 W_i 的子文件标记为$W_{i,\ f}$： $\forall f \in \mathcal{F}$（$W_{i,\ f}$ 假定取自某个阿贝尔群），其中 $\mathcal{F}$ 是一个大小为 F的集合。集中式编码缓存框架包含两个阶段：放置阶段 和交付阶段。放置阶段在非高峰时段进行。在放置阶段，利用通信信道将文件中的一些子文件存入各客户端的缓存，使缓存得到充分填充。交付阶段在高峰时段进行。在交付阶段（高峰时段），用户 u 请求特定文件 W_{d_u}，其中 $d_u \in [N]$。传输方案通过发送 W_{d_u} 的子文件 $u \in \mathcal{U}$ 的编码传输来满足这些请求。编码缓存方案的速率 R 定义为服务器传输的比特数与每个文件的大小（以比特计）之比。当传输方案为线性时，也就是说，服务器的每次传输都是 W_{d_u} 的子文件 $u \in \mathcal{U}$的线性组合，则速率可计算为

$$\text{Rate } R = \frac{\text{Number of bits transmitted by server}}{\text{Size of the file in bits}} = \frac{\text{Number of transmissions in the delivery phase}}{F}.$$

我们将介绍由 [3]提出的适用于此设置的新型编码缓存方案。为介绍我们的编码缓存方案，我们定义缓存矩阵，它本质上刻画了编码缓存方案的缓存阶段。

定义 2 (缓存矩阵**)**。 考虑一个二进制矩阵 C ，其行由一个大小为 K的集合 $\mathcal{U}$ 索引，列由一个大小为 F的集合 $\mathcal{F}$ 索引，并且每一行中1的数量相同（设为 Z）。那么，矩阵 C定义了一个缓存方案，该方案有 K 个用户（由 $\mathcal{U}$索引），子分组化为 F （由 $\mathcal{F}$索引），未缓存比例$(1 - \frac{M}{N}) = \frac{Z}{F}$，具体如下：

— —

- 用户 $u \in \mathcal{U}$缓存$W_{i,\ f}$： $\forall i \in [N]$若 $C(u, f) = 0$，则缓存该子文件；若 $C(u, f) = 1$，则不缓存该子文件。

我们随后将矩阵 C称为$(K, F, (1 - \frac{M}{N}))$缓存矩阵。

如果子文件$W_{i,\ f}$在用户 u 的缓存中不可用，则称该子文件缺失。为了构造传输方案，我们先描述一种基于上述矩阵缓存方案的传输方式，该方式将服务多个用户。

引理 1. 考虑 C 的一个单位子矩阵，其行由 $\{u_1, u_2, .., u_l\colon\ u_i \in \mathcal{U}\}$ 给出，列由 $\{f_1, f_2, .., f_l\colon f_i \in \mathcal{F}\}$ 给出，使得 $C(u_i, f_i) = 1, \forall i \in [l]$，而 $C(u_i, f_j) = 0, \forall i, j \in [l]$，其中 $i \neq j$。对于每个 $i \in [l]$，子文件$W_{d_{u_i},\ f_i}$ 在用户 u_i 处不可用，但可通过传输 $\sum_{i=1}^{l} W_{d_{u_i},\ f_i}$ 解码。

证明： 根据单位子矩阵的定义，对于每个 $i \in [l]$，子文件 $W_{d_{u_i},\ f_i}$ 在用户 u_i 处不可用，但在用户 $\{u_1, u_2, .., u_l\} \backslash u_i$ 处可用。因此，每个用户 $u_i\colon i \in [l]$ 都可以通过传输 $\sum_{i=1}^{l} W_{d_{u_i},\ f_i}$ 解码子文件 $W_{d_{u_i},\ f_i}$，该子文件在其缓存中不可用。 ■

现在我们说明如何利用 C 的单位子矩阵覆盖来构造传输方案。

定理 1。 考虑缓存矩阵 C的一个单位子矩阵覆盖 $\mathfrak{C} = \{C_1, C_2, .., C_S\}$。那么，根据引理 *1*，与 $C_i\colon$ $i \in [S]$ 对应的传输，是由 C 定义的缓存方案的有效传输方案（即该方案满足所有用户需求），且该传输方案的速率为 $R = \frac{S}{F}$。

—

证明：任取用户 u缺失的一个子文件 $W_{d_u,\ f}$。那么 $C(u,\ f)\ =1$，并且矩阵 C中的这个元素至少会被一个单位子矩阵覆盖，设该单位子矩阵为 C_i，它属于 $\mathfrak{C}$，因为 $\mathfrak{C}$ 是 C的一个单位子矩阵覆盖。根据引理 1，与单位子矩阵 C_i 对应的传输将确保子文件 $W_{d_u,\ f}$ 由对应的用户 u 解码，该用户正缺失此子文件。因此，与 $C_i\in\mathfrak{C}$对应的传输能够实现对任意缺失子文件的解码。由于 $\mathfrak{C}$ 中的单位子矩阵数量为 S，该传输方案的速率为 $R=\frac{S}{F}$。

– ■

III. 组合设计背景

在上一节中，我们为缓存问题建立了一个二元矩阵模型。在接下来的几节中，我们将使用组合设计和子空间设计来构造缓存矩阵。为此，我们首先回顾组合设计及其构造的一些基本定义。有关组合设计的更多详情，请参阅 [38], [39]。随后，我们回顾子空间设计的一些相关方面，它们是组合设计的 q类比。

定义 3 (设计 $(\mathcal{X},\mathcal{A})$)。设计是一个元素对 $(\mathcal{X},\mathcal{A})$，满足以下性质：*(D1).* $\mathcal{X}$ 是一个称为点的元素集合；*(D2).* $\mathcal{A}$ 是 $\mathcal{X}$ 的非空子集的一个集合（即多重集），这些子集称为区组。

现在定义 t-设计。

定义 4 (t-设计)。设 v、k、λ 和 t 为满足 $v>k\geq t$ 的正整数。t-(v,k,λ)-设计（或简称t-设计）是满足以下性质的设计 $(\mathcal{X},\mathcal{A})$：

(T1). $|\mathcal{X}|=v$，*(T2).* 每个区组恰好包含 k 个点，并且 *(T3).* 每个由 t个不同点组成的集合恰好包含在 λ 个区组中。

考虑一个非空的 $Y\subseteq\mathcal{X}$，使得 $|Y|=s\leq t$。那么恰好有

$$\lambda_s=\lambda\frac{\binom{v-s}{t-s}}{\binom{k-s}{t-s}} \tag{1}$$

$\mathcal{A}$ 中包含 Y中的所有点的区组。还可以证明，$b=\lambda_0=\lambda\frac{\binom{v}{t}}{\binom{k}{t}}$ 是t-设计中区组的数量。 —

例1. *[参数化构造]*一个具有 $\lambda=1$ 的 t（即 t-$(v,k,1)$ 设计）称为 ***Steiner***系统，其存在性将在 *[40]*中讨论。文献 *[41]*介绍了 $t=3$ 和 $t=4$ 的 ***Steiner***系统构造。其他 ***Steiner***系统的一般构造可见 *[39]*。这里我们采用一种特定的构造。

- 文献 *[38]*, 提出了一种参数为 $t=3$, $v=q^2+1$, $k=q+1$ 的***Steiner***系统构造，其中 q是满足 $q\geq 2$的素数幂。

在下文的示例以及本文中的其他一些地方，我们在明确写出区组时省略圆括号和逗号（例如，区组 $\{l,m,n\}$ 写作 lmn）。

例 2. 考虑以下集合 $\mathcal{X}$及其子集的相关集合 $\mathcal{A}$。 $\mathcal{X}=\{1,2,3,4,5,6,7,8\}$和 $\mathcal{A}=\{1256,3478,1357,2468,1458,2367,1234,5678,1278,3456,1368,2457,1467,2358\}$。不难验证， $\mathcal{X}$ 中的每个大小为3的子集都恰好出现在一个区组中（即 $\mathcal{A}$中的一个子集）。因此，这是一个3-$(8,4,1)$ 设计（*Steiner*系统）。

定义5（平衡不完全区组设计）。 t-设计中具有 $t=2$ 的称为平衡不完全区组设计（*BIBD*），记作 *(v, k, λ)-BIBD*。

由 **(1)** 可知，每个 (v, k, λ)-BIBD 恰有 $b=\frac{vr}{k}$ 个区组，并且包含每个点的区组数量恰好为

$$r=\frac{\lambda(v-1)}{k-1}. \tag{2}$$

例3。 设 $\mathcal{X}=\{1,2,3,4,5,6,7,8,9\}$ 和 $\mathcal{A}=\{357,123,456,789,147,258,369,159,267,348,168,249\}$.。区组数量为 $b=12$，且 $\mathcal{X}$ 中的每个元素恰好出现在 $r=4$ 个区组中。另请注意，每一对元素在 $\mathcal{A}$中都恰好出现在一个区组中。因此，（$\mathcal{X}$, $\mathcal{A}$） 是一个 *(9,3,1)-BIBD* 的例子。

现在定义对称平衡不完全区组设计。

定义6（对称平衡不完全区组设计）。 满足 $b=v$ （或等价地，满足 $r=k$）的 v, k, λ)-平衡不完全区组设计称为对称平衡不完全区组设计。

如 [38], 所述，在对称平衡不完全区组设计中，任意两个区组 A_1， $A_2\in\mathcal{A}$

$$|A_1\cap A_2|=\lambda. \tag{3}$$

例4。 *[参数化构造]* 下面给出文献中已知的一些平衡不完全区组设计构造：

- 参数为 $v=n^2+n+1$, $k=n+1$, $\lambda=1$ 的对称平衡不完全区组设计是在 *[38]* 中利用阶为 n 的射影平面构造的，其中 n 是满足 $n\geq 2$ 的素数幂。
- 参数为 $v=n^2$, $k=n$, $\lambda=1$ 的平衡不完全区组设计是在 *[38]* 中利用阶为 n 的仿射平面构造的，其中 n 是满足 $n\geq 2$ 的素数幂。
- 文献 *[42]*提出了一种利用仿射可分解平衡不完全区组设计构造对称平衡不完全区组设计的方法。

现在我们定义横截设计。

定义7（横截设计）。 设横截设计的阶或组大小为 n、区组大小为 k、指标为 λ，记作 TD_λ（k、 n），它是一个三元组（$\mathcal{X}$、 $\mathcal{G}$、 $\mathcal{B}$），其中

(TD1)。 $\mathcal{X}$ 是一个包含 kn 个元素的集合。*(TD2)*。 $\mathcal{G}$ 是 $\mathcal{X}$ 的一个划分，分为 k 个集合（即组），每个集合包含 n个元素。*(TD3)*。 $\mathcal{B}$ 是 $\mathcal{X}$ 的一个由大小为 k的子集组成的集合（即区组）。*(TD4)*。 $\mathcal{X}$ 中的任意一对元素，要么恰好包含在一个组中，要么恰好包含在 λ 个区组中，但不会同时属于两者。

其中 $\lambda = 1$ 的横截设计记作 TD(k, n)。由上述性质可知， $|\mathcal{B}|= n^2$，并且 $\mathcal{X}$ 中的每个元素，对于 $\lambda = 1$ [43]都会出现在 n 个区组中。

例5。 如下给出了一个 *TD*$(4,\ 3)$ 设计。考虑集合 $\mathcal{X} = \{1,2,3,4,5,6,7,8,9,10,11,12\}$、 $\mathcal{G} = \{\{1,2,3\},\{4,5,6\},\{7,8,9\},\{10,11,12\}\}$ 和
$\mathcal{B} = \{\{1,4,7,10\},\{1,5,8,11\},\{1,6,9,12\},\{2,4,9,11\},\{2,5,7,12\},\{2,6,8,10\},\{3,4,8,12\},\{3,5,9,10\},$
$\{3,6,7,11\}\}$. 可以验证， $\mathcal{X}$ 中的每一对元素恰好出现在一个组或一个区组中，但不会同时出现在两者中。

例6. *[参数化构造]* 文献中已知的一些 *TD*(k, n) 构造如下：

- 参数为 $\lambda = 1, k = q, n = q$ 的横截设计可利用正交数组构造，其中q 是满足 $q \geq 2$ 的素数幂。

- 参数为 $\lambda = 1, k = q + 1, n = q$ 的横截设计可利用正交数组构造，其中 q 是满足 $q \geq 2$ 的素数幂。

我们在本文中考虑的最后一种组合结构是子空间设计的概念，它是组合设计的 q-模拟。此处我们回顾这些对象的一些基本定义。关于这些对象的更多细节可见于 [44]–[46]。

定义 8 (子空间设计)。 设 $\mathcal{V}$ 是有限域 $\mathbb{F}_q$ 上的一个维数为 v 的向量空间。维数为 k 的子空间称为 k维子空间。设计的 *q*-模拟定义如下。设 $0 \leq t \leq k \leq v$为整数， λ 为非负整数。若元素对 $\mathcal{D} = (\mathcal{V}, \mathcal{A})$中， $\mathcal{A}$ 是 $\mathcal{V}$的一个由 k维子空间（区组）组成的集合，则称其为 $\mathcal{V}$上的一个 t-$(v, k, \lambda)_q$-子空间设计，若 $\mathcal{V}$ 的每个 t 维子空间恰好包含于 λ个区组中。

一个 t-(v， k，1) 子空间设计也称为等价 Steiner 系统的 q-类比。下面回顾一些已知的子空间设计构造。

Ex例 7. 这里回顾文献中已知的几种子空间设计构造。

1) 对于任意 $0 \leq t \leq k \leq v$，以及任意为素数幂的 q， $\mathbb{F}_q^v$中所有 k 维子空间构成一个 $t-(v, k, \left\langle v-t, k-t\right\rangle)_q$ 子空间设计。具体而言，当 $t = k$ 时，我们得到一个 $k-(v, k, 1)_q$ 子空间设计。*2)* 对于任意素数幂 q，文献 *[44]* 证明，存在一个 $1-(v, k, 1)_q$ 设计，当且仅当 k 整除 v。*3)* 文献 *[47], [48],* 给出了一种非平凡 $2-(v, 3, q^2+q+1)_q$ 设计的构造，其中 q 为任意素数幂，且 $v \geq 7$ 满足 v 与整数 *24* 互素。*4)* 文献 *[49]* 的作者证明，当 n 足够大时，对于任意 t 和任意 $k > 12(t+1)$，在 $\mathbb{F}_q$ 上都存在子空间设计。*5)* 针对特定参数，还有许多其他单独的构造，见文献 *[44],* ；本文使用其中一些构造，给出由子空间设计得到的编码缓存方案的数值示例。

在一些构造中，我们将使用以下设计关联矩阵的思想 .

定义 9（关联矩阵）。设 $(\mathcal{X}, \mathcal{A})$ 是一个设计，其中 $\mathcal{X} = \{x_1, ..., x_v\}$ 和 $\mathcal{A} = \{A_1, ..., A_b\}$。$(\mathcal{X}, \mathcal{A})$ 的关联矩阵是 $v \times b$ 二进制矩阵 $M =$（M（i, j），其定义规则如下：

$$M(i,j) = \begin{cases} 1, & \text{if } x_i \in A_j, \\ 0, & \text{if } x_i \notin A_j. \end{cases}$$

例 8。由 $\mathcal{X} = \{1, 2, 3, 4, 5, 6, 7\}$ 和 $\mathcal{A} = \{127, 145, 136, 467, 256, 357, 234\}$ 给出的（7，3，1）-平衡不完全区组设计的关联矩阵如下所示。

$$C = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 127 & 145 & 136 & 467 & 256 & 357 & 234 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \end{array}.$$

IV. 基于设计构造的新型编码缓存方案汇总

表 I 汇总了后续各节将从不同设计构造的编码缓存方案所涉及的所有缓存参数。参数 v，k，t 基于用于构造这些方案的设计参数。表 I 还在最后两列列出了 Ali-Niesen 方案 [3]的（最优）速率 R^* 以及相应的子包化级别 F^*（已证明，对于在 [37] 中达到最优速率 R^* 的所有基于 PDA 的方案而言，该级别均为最优）。

将表 I 的结果应用于第 III 节给出的设计参数化构造，我们得到以下关于本文所获得并列于第 IV-A 节中的特定构造的结果。第 XIII 节展示了比较其中许多特定构造与现有基准方案的数值结果。

组合的 组合设计（$\lambda = 1$）	$(1 - \frac{M}{N})$	K	F	R	$R^*(=(\frac{K(1-\frac{M}{N})}{1+K\frac{M}{N}}))$	$F^*(=\binom{K}{KM/N})$
BIBD	$\frac{\frac{k}{t}}{\binom{v}{t}k(v-t+1)}$	v	$\frac{v(v-1)}{k(k-1)}$	$\frac{k(k-1)}{v}$	$\frac{k}{1+v-k}$	$\binom{v}{k}$
t-设计	$\frac{\binom{k}{t}}{\binom{v}{t}k}$	$\binom{v}{t-1}$	$\frac{\binom{v}{t}k}{\binom{k}{t}}$	$\frac{\binom{k-1}{t-1}\binom{k}{t}v}{\binom{v}{t}k}$	$\frac{\binom{k-1}{t-1}}{1+\binom{v}{t-1}-\binom{k-1}{t-1}}$	$\binom{\binom{v}{t-1}}{\binom{k-1}{t-1}}$
横截设计	$\frac{\frac{1}{n}}{\langle v-t+1,1\rangle q^{t-1}}$	n^2	kn	1	$\frac{n}{1-n+n^2}$	$\binom{n^2}{n}$
子空间设计	$\frac{\langle v-t+1,1\rangle q^{t-1}}{F}$	$\langle v, t-1\rangle$	$\frac{\langle v,t\rangle\langle k,1\rangle}{\langle k,t\rangle}$	$\frac{\langle k-1,t-1\rangle\langle v,1\rangle q^{t-1}}{F}$	（太大，无法放入）	（太大，无法放入）

表 I：本文基于编码设计的编码缓存方案的参数及其下界 [3]。表中未列出的某些条目可利用顶行中的 R^* 和 F^* 表达式轻松计算得出。

A. 基于现有设计的编码缓存方案的特定构造

1) 平衡不完全区组设计： 例4中所述构造的传输方案（见第 V 节）的参数如下：

a) 参数为 $v = n^2 + n + 1$、 $k = n + 1$、 $\lambda = 1$ 的对称平衡不完全区组设计将给出一个参数为 $F = n^2 + n + 1$、 $K = n^2 + n + 1$，速率为 $= 1$，$(1 - \frac{M}{N}) = \frac{n+1}{n^2+n+1}$ 的编码缓存方案。

b) 参数为 $v = n^2$、 $k = n$、 $\lambda = 1$ 的平衡不完全区组设计将给出一个参数为 $F = n^2 + n$、$K = n^2$，速率为 $= \frac{n}{n+1}$，$(1 - \frac{M}{N}) = \frac{1}{n}$ 的编码缓存方案。

注意，对于上述两个方案，我们有 $F = O(K)$，且 $1 - \frac{M}{N} = \Theta(\frac{1}{\sqrt{K}})$，而 $R \leq 1$.

*2) Steiner*系统：对于例1中所述的构造，第VI节所介绍的传输方案的参数如下。

a) 参数为 $t = 3,\ v = q^2 + 1,\ k = q + 1,\ \lambda = 1$ 的 t-设计，将给出一个参数为 $F = (q^2 + 1)(q + 1)$, $K = (q^2+1)_2$ $^{q^2}$、速率为 $= (q-1)\,2(q+1)$、$(1 - \frac{M}{N}) = (q-1)\,q(q^2+1)$ 的编码缓存方案。

注意，对于上述编码缓存方案的构造，我们有 $F = O(K^{\frac{3}{4}})$ $1 - \frac{M}{N} = \Theta(\frac{1}{\sqrt{K}})$ 和 $R \leq 1$.

3) 横截设计： 例6 中所述构造对应的传输方案（见第 VII 节）的参数如下：

a) 参数为 $\lambda = 1,\ k = q,\ n = q$ 的横截设计，将给出一个参数为$F = q^2$, $K = q^2$、速率为 $= 1$、$(1 - \frac{M}{N}) = \frac{1}{q}$ 的编码缓存方案。

b) 一个参数为 $\lambda = 1,\ k = q + 1,\ n = q$ 的横截设计，将给出一个参数为 $F = q^2 + q$, $K = q^2$、速率为 $= 1$ 的编码缓存方案，$(1 - \frac{M}{N}) = \frac{1}{q}$。

对于上述两种构造，我们都有 $F = O(K)$，且 $(1 - \frac{M}{N}) = O(\frac{1}{\sqrt{K}})$。

4）子空间设计：对于任意素数幂 q，我们在例7中指出，存在一个 k–$(v, k, 1)_q$ 子空间设计（其区组为 v 维空间中所有 k 维子空间的集合，该空间定义在 $\mathbb{F}_q$ 上）。这将导出一个编码缓存方案（见第八节），其参数为 $K = \langle v, k-1 \rangle$, $F = \langle v, k \rangle \langle k, 1 \rangle$, $1 - M/N = \frac{\langle v-k+1, \quad 1 \rangle q^{k-1},,}{\langle v\ k \rangle \langle k\ 1 \rangle}$ 以及 $R = \frac{\langle v, \quad 1 \rangle q^{k-1},,1 \rangle}{\langle v\ k \rangle \langle k}$。由于这些参数之间相互影响，很难确定该方案的渐近性质。有趣的是，从数值结果（第 XIII 节）中可以看到，由这些子空间设计（以及文献中其他子空间设计的若干实例）得到的缓存方案，在本文设计的方案中具有较小的缓存占比（即较大的 $1 - M/N$ 值）。

在接下来的各节中，我们将基于上述组合设计和子空间设计构造缓存方案。每种构造都取决于相应类型设计的存在性。在每种情况下，我们都利用给定的设计定义缓存矩阵，并根据设计参数求得其参数 K、F，以及 $(1 - \frac{M}{N})$。随后，我们根据设计的性质定义并证明该缓存矩阵的单位子矩阵覆盖。以下各项构造中，单位子矩阵覆盖的证明格式相同，现依次说明如下：

1) 我们介绍一种从缓存矩阵中选取子矩阵的方法，并通过以下方式证明该子矩阵是单位子矩阵。

a) 我们证明该子矩阵的行数和列数相等。

b) 接下来证明该子矩阵的每一行和每一列的权重均为 1。

2) 接下来证明所选取的单位子矩阵彼此不重叠。

3) 最后证明缓存矩阵中的所有 1 都被这些单位子矩阵的集合覆盖，从而证明该集合构成一个单位子矩阵覆盖。

V. 基于平衡不完全区组设计（其中 $\lambda = 1$）的编码缓存方案

考虑一个 $(v, k, 1)$-平衡不完全区组设计 $(\mathcal{X}, \mathcal{A})$，其中 $|\mathcal{A}| = b$ 为区组的数量。我们以任意方式对 $\mathcal{X}$ 中的元素排序。对于不同的 $x, y \in \mathcal{X}$，若 x 在 $\mathcal{X}$ 的排序中位于 y 之前，则称 $x < y$。设区组 $A \in \mathcal{A}$ 中的元素依次记为 $\{A(0)\ A(1) \ldots, A(k-1)\}$，其中 $A(0) < A(1) < .. < A(k-1)$。令 C 表示 $(v, k, 1)$-平衡不完全区组设计的关联矩阵。

注 1。注意，根据（2），$\mathcal{X}$ 中的每个元素都出现在 $r = \frac{v-1}{k-1}$ 个区组中。因此，关联矩阵 C 的每一行权重均为 r。此外，每个区组 $A \in \mathcal{A}$ 的大小为 k，因此矩阵 C 的列权重为 k。所以，由 *BIBD* 构造的二进制缓存矩阵具有恒定的行权重和列权重。

现在，$(1 - \frac{M}{N}) = \frac{r}{b} = \frac{k}{v}$（因为 $vr = bk$），并且 $F = b = \frac{v\ (v-1)\ k\ (k-1)}{}$。因此，设计（$\mathcal{X}$，$\mathcal{A}$）的关联矩阵 C 将给出一个 $\left(v, \frac{v}{k}\frac{(v-1)\ \ k-1)}{}\ \ \frac{k}{v}\right)$-缓存矩阵。

对于 $x \in \mathcal{X}$，令 $B_x \triangleq \{A \in \mathcal{A} : x \in A\}$。注意，根据具有 $\lambda = 1$ 的 BIBD 的性质，有 $|B_x| = r$。在下一个引理中，我们描述矩阵 C 的一个单位子矩阵。

引理 2. 对于任意 $x \in \mathcal{X}$，记 B_x 为 $B_x = \{A_1, ..., A_r\}$，其中 $x = A_i(j_i)$：$j_i \in \{0, 1, \ldots, k-1\}$，即 x 是区组 A_i 中的第 j_i^{th} 个元素。考虑 C 的子矩阵 C_x，其列由 B_x 索引，行由 $\{A_i(j_i \oplus_k 1) : i \in [r], A_i \in B_x\}$ 索引。则 C_x 是 C 的一个大小为 $r = \frac{v-1}{k-1}$ 的单位子矩阵。

证明：注意，C 的子矩阵 C_x 有 r 列。现在我们证明它有 r 行。由于 $(\mathcal{X}, \mathcal{A})$ 是一个参数为 $\lambda = 1$ 的平衡不完全区组设计，元素 $\{x, A_i(j_i \oplus_k 1)\} \subset A_i$ 仅属于该区组，不属于任何其他 $A_{i'}$，对任意 $i' \neq i$ 均如此。因此，对任意 $i' \neq i$，元素 $A_i(j_i \oplus_k 1) \neq A_{i'}(j_{i'} \oplus_k 1)$。故 C_x 中有 r 行。考虑由 $A_i(j_i \oplus_k 1)$ 索引的行。再次利用性质 $\lambda = 1$，在 C_x 中，由 $A_i(j_i \oplus_k 1)$ 索引的行里，唯一对应元素为 1 的列索引是 A_i，不存在其他列。因此，C_x 的每一行在某一列中恰有一个元素 1。现在考虑 C_x 的任意一列，例如由 A_i 索引的列。由于 $A_{i'}(j_{i'} \oplus_k 1)$ 仅当 $i = i'$ 时成立，因此列 A_i 仅在行 $A_i(j_i \oplus_k 1)$ 中有一个 1。故 C_x 是 C 的一个大小为 r 的单位子矩阵。

■

在接下来的两个引理中，我们将证明单位子矩阵 C_x：$x \in \mathcal{X}$ 之间没有重叠，并且这些单位子矩阵将覆盖矩阵 C 中满足 $C(x, A) = 1$ 的所有条目。

引理 3. 对于不同的 x_1、$x_2 \in \mathcal{X}$，不存在满足 $C(x, A) = 1$ 的 $x \in \mathcal{X}$、$A \in \mathcal{A}$，使得 $C(x, A)$ 同时被 C_{x_1} 和 C_{x_2} 覆盖，其中 C_{x_1} 和 C_{x_2} 的定义见引理 *2*。

证明：假设 $C(x, A) = 1$ 同时被 C_{x_1}, C_{x_2} 覆盖。根据 C_{x_1} 和 C_{x_2} 的定义，这意味着 $x_1, x_2 \in A$。设元素 x_1 和 x_2 分别出现在 A 的 j 和 j' 位置，其中 $j \neq j'$。因此，根据我们的构造，$x = A(j \oplus_k 1) = x = A(j' \oplus_k 1)$。然而，$j \neq j'$。这就产生了矛盾。因此，不存在 $C(x, A) = 1$ 同时被 C_{x_1} 和 C_{x_2} 覆盖。

■

引理 4 。矩阵集合 $\{C_x : x \in \mathcal{X}\}$ 构成一个不重叠单位子矩阵覆盖 C.

证明：矩阵 C 中 1 的总数等于每行中 1 的数量与行数的乘积，因此等于 rv。对于每个 $x \in \mathcal{X}$（注意 $|\mathcal{X}| = v$），都存在一个单位子矩阵 C_x。由引理 2 和 3 可知，每个单位子矩阵的大小为 r，且这些单位子矩阵两两不重叠。这些单位子矩阵将覆盖矩阵 C 中的 vr 个 1，其数量等于 C中 1 的总数。因此，$\{C_x : x \in \mathcal{X}\}$ 构成 C的不重叠单位子矩阵覆盖。

■

因此，我们有以下定理来概述该缓存方案。

定理 2。*$(v, k, 1)$*-平衡不完全区组设计的关联矩阵构成一个 $\left(K = v, F = \binom{v}{k}_{(v-1)}, \binom{(k-1)}{(1-\frac{M}{N})} = \binom{k}{v}\right)$ 缓存矩阵。此外，存在一种速率为 $R = \binom{k}{(k-1)}_{v-1}$ 的传输方案。

证明：缓存矩阵 C 的参数已定义。根据引理 2、3 和 4，我们得到一个由 v 个单位子矩阵构成的单位子矩阵覆盖。因此，根据定理 1，$S = v$，且速率为 $R = \frac{S}{F} = \binom{k}{(k-1)}_{v-1}$。

■

例 9。考虑例 *8* 中给出的(7、3、1)-平衡不完全区组设计。我们描述 $\mathcal{X}$中与元素‘1’对应的单位子矩阵 C_1（根据引理 *2*）。元素‘1’出现在区组 $\{127, 145, 136\}$中。于是，单位子矩阵矩阵 C_1 的列由127、145、136 索引，行由2、4、3 索引（因为在区组127、145、136 中，元素‘1’之后出现的下一个元素分别是2、4、3），矩阵为 C。类似地，$\mathcal{X}$ 中与元素‘6’对应的单位子矩阵C_6 在矩阵 C中的列索引和行索引分别为136、467、256 以及1、7、2 。如此一来，子矩阵 $C_i : i \in \mathcal{X}$为我们提供了速率为1 的传输方案。

VI. 基于 t-设计的编码缓存方案

现在，我们通过由 t-设计导出的二进制矩阵来描述一种编码缓存方案。

设 $(\mathcal{X}, \mathcal{A})$ 表示一个 t-$(v, k, 1)$ 设计。设该 t-设计中的区组记为 $\mathcal{A} = \{B_1, ..., B_b\}$，其中 $b = \binom{v}{t} \binom{k}{t}$。我们按如下方式构造二进制矩阵 T。令 T 的行由 $\mathcal{X}$ 的所有 $(t-1)$ 元子集索引。令列由 $\{(y, B) : y \in B, B \in \mathcal{A}\}$ 索引。矩阵 T 的行数为 $\binom{v}{t-1}$。矩阵 T 的列数为 $bk = \binom{v}{t} k \binom{k}{t}$。对于某个 $D \in \binom{\mathcal{X}}{t-1}$，矩阵 $T = (T(D, (y, B)))$ 按如下规则定义：$T(D, (y, B)) = \begin{cases} 1, & \text{若 } D \cup \{y\} \subset B \ |D \cup \{y\}| = t, \text{否则为} \\ 0 & . \end{cases}$

注 2. 根据 t-设计的性质，很容易计算出 T 的每一行中 1 的数量为 $\lambda_{t-1}\binom{k-t+1}{1} = (v - t + 1)$。此外，每一列中 1 的数量为 $\binom{k-1}{t-1}$。因此，二进制缓存矩阵 T也是一个行列常重矩阵。

矩阵 T 给出了一个 $\left(\binom{v}{t-1},\ \binom{v}{t}k\binom{k}{t}\ ,\ \binom{k}{t}(v-t+1)\binom{v}{t}k\right)$-缓存矩阵。在下一个引理中，我们将描述矩阵 T的一个单位子矩阵。为此，我们需要定义几个集合。对于某个 $y \in \mathcal{X}$，定义 $\mathcal{B}_y \triangleq \{B \in \mathcal{A}: y \in B\}$，即包含 y的区组集合。由 (1)，$|\mathcal{B}_y| = \lambda_1 = \binom{v-1}{t-1}\binom{k-1}{t-1}$。将 $\mathcal{B}_y$记为 $\mathcal{B}_y = \{B_1, \ldots, B_{\lambda_1}\}$。对于任意 B_i，将 $\{D_{i,j}: j \in [\binom{k-1}{t-1}]\}$ 记为集合 $\binom{B_i \backslash y}{t-1}$，即 $B_i \backslash y$的所有大小为 $(t-1)$ 的子集。

引理 5。对于某些 $j \in [\binom{k-1}{t-1}]$ 和 $y \in \mathcal{X}$，考虑 $T_{y,\ j}$ 的子矩阵，其行由 T 的$\{D_{i,j}: \forall i \in [\lambda_1]\}$ ($D_{i,\ j}$ 如上所定义）索引，列由 $\{(y、B_i): B_i \in \mathcal{B}_y\}$索引。则 $T_{y,\ j}$ 是 T 的一个大小为 $\lambda_1 = \binom{v-1}{t-1}\binom{k-1}{t-1}$ 的单位子矩阵。

证明：显然，$T_{y,\ j}$ 中的列数量为 λ_1。首先注意到，$\{D_{i,j}$ 中的行：$\forall i \in [\lambda_1]\}$ 均不相同，即对于 $i \neq i'$，$D_{i,\ j} \neq D_{i',\ j}$。否则，注意到 $D_{i,\ j} \cup y = D_{i',\ j} \cup y \in B_i \cap B_{i'}$。但这与以下事实矛盾：$\mathcal{X}$ 的任意大小为 t的子集只出现在一个区组中。因此，$|\{D_{i,j}: i \in [\lambda_1]\}| = \lambda_1 = \binom{v-1}{t-1}\binom{k-1}{t-1}$，根据（1），从而$T_{y,\ j}$ 是一个大小为 λ_1的方阵。

现在考虑 $i \in [\lambda_1]$的一行 $T_{y,\ j}$，其索引为 $D_{i,\ j}$。假设某个 $B \in \mathcal{B}_y$ 对应的列，其索引为 $(y,\ B)$，在索引为 $D_{i,\ j}$ 的行中取值为 1。这意味着 $D_{i,\ j} \cup \{y\} \subset B$。但恰好存在一个区组 B，使得 $D_{i,\ j} \cup \{y\} \subset B$，而这恰好就是 $B = B_i$（因为 $\lambda = 1$）。因此，$T_{y,\ j}$ 的每一行中恰好只有一个元素为 1。

现在考虑 $T_{y,\ j}$ 的一列，其索引为 $(y,\ B_i)$，对应某个 $B_i \in \mathcal{B}_y$。假设对于某个 $D_{i',\ j}$，索引为 $D_{i',\ j}$ 的行在索引为 $(y,\ B_i)$ 的列中取值为 1。那么必有 $D_{i',\ j} \subset B_i \backslash y$，因此 $D_{i',\ j} \cup \{y\} \subset B_i$。同样，根据 $\lambda = 1$ 的 t-设计的性质，可知 $i = i'$（否则 $D_{i',\ j} \cup \{y\} \in B_i \cap B_{i'}$，这就产生了矛盾）。因此，$T_{y,\ j}$ 的每一列中恰好只有一个元素为 1。这就证明了该引理。

■

在接下来的两个引理中，我们将证明这些单位子矩阵之间没有重叠，并且这些单位子矩阵将覆盖矩阵 T 中 $T(D,\ (y,\ B)) = 1$ 所在的所有条目。

引理 6. </style}任何 $(D,\ (y,\ B)) = 1$ 只要满足 $y \in \mathcal{X}$、$B \in \mathcal{A}$、$D \in \binom{\mathcal{X}}{t-1}$，就恰好会被 T 的一个单位子矩阵覆盖（定义见引理 5）。

证明：设 $T(D,\ (y,\ B)) = 1$ 被 T 的一个单位子矩阵 $T_{y'、\ j}$ 覆盖。由于 $T(D,\ (y,\ B)) = 1$，我们有 $D \cup \{y\} \subset B$。根据引理 5 中 $T_{y'、\ j}$ 的定义，首先必须有 $y = y'$。此外，对于某个包含 y 的 B_i，必有 $D \subset B_i \backslash y$。因此，$\{y\} \cup D \subset B_i$，这意味着 $B_i = B$（因为 $\lambda = 1$）。所以，覆盖 $T(D,\ (y,\ B)) = 1$ 的唯一传输是 $T_{y、\ j}$，其中 j 满足 $D = D_i$，且 j 是 $\binom{B \backslash y}{t-1}$ 中唯一的 j^{th} 集合。

■

引理 7. 矩阵集合 $\{T_{y,j}: \forall y \in \mathcal{X},\ \forall j \in [\binom{k-1}{t-1}]\}$ 构成了 T 的不重叠单位子矩阵覆盖。

证明：T 中 1 的总数等于每行中 1 的数量与行数的乘积，因此等于 $(v-t+1)\binom{v}{t-1} = t\binom{v}{t}$。每个单位子矩阵的大小为 $\binom{v-1}{t-1}\binom{k-1}{t-1}$。由于对于每个 $j \in [\binom{k-1}{t-1}]$，$y \in \mathcal{X}$，我们都有一个单位子矩阵 $T_{y,\ j}$，因此单位子矩阵的数量为 $= \binom{k-1}{t-1}v$。此外，根据引理 6，这些单位子矩阵之间没有重叠。因此，所有单位子矩阵所覆盖的 1 的总数为

所有单位子矩阵 $= \binom{k-1}{t-1}\binom{v-1}{t-1}v\binom{k-1}{t-1} = \binom{v-1}{t-1}v = t\binom{v}{t}$，这等于矩阵 T中 1 的总数。因此， $\{T_{y,j}: \ y \in \mathcal{X}, \ j \in [\binom{k-1}{t-1}]\}$ 构成了 T的一个不重叠单位子矩阵覆盖。

■

因此，我们得到以下总结该缓存方案的定理。

定理 3. 一个 t-$(v,k,1)$-设计的矩阵 T构成一个 $\left(K = \binom{v}{t-1}, F = \binom{v}{t}k\binom{k}{t}^{-1}(1-\frac{M}{N}) = \binom{k}{t}(v-t+1)\binom{v}{t}k\right)$-缓存矩阵。此外，还存在一个速率为 $R = \binom{k-1}{t-1}\binom{k}{t}v\binom{v}{t}k$ 的传输方案。

证明： 缓存矩阵 T 的参数已定义。根据引理 5、6 和 7，我们得到一个单位子矩阵覆盖，其中包含 $\binom{k-1}{t-1}v$ 个单位子矩阵。因此，根据定理 1， $S = \binom{k-1}{t-1}v$ 且速率为$R = \frac{S}{F} = \binom{k-1}{t-1}\binom{k}{t}v\binom{v}{t}k$ 。

■

例 *10.* 考虑例 *2* 中给出的 *3-(8, 4, 1)*设计。现在我们描述一些单位子矩阵，即 $T_{4,}$*1*、$T_{4,}$*2* 和 $T_{4,}$*3*。包含元素 *'4'*的区组集合记为 $\mathcal{B}_4 = \{3478, 2468, 1458, 1234$*3456*、*2457*、$1467\}$。显然，根据我们的构造， T的行由 $\mathcal{X}$ 的 $(t-1)$ 个 $= 2$ 大小的子集索引。单位子矩阵 $T_{4,}$*1*、$T_{4,}$*2* 和 $T_{4,}$*3* 的行分别由 $\{37, 26, 15, 12, 35, 25, 16\}$、$\{38, 28, 18, 13, 36, 27, 17\}$ 和 $\{78, 68, 58, 23, 56, 57, 67\}$ 索引，列则由矩阵 T 的 $\{(4, B_i): \ B_i \in \mathcal{B}_4\}$ 索引。这样，我们可以利用矩阵 $T_{y,j}: \forall y \in \mathcal{X}, \ \forall j \in [\binom{k-1}{t-1}]$，得到 T 的单位子矩阵覆盖，从而构造出速率为 $= \frac{3}{7}$ 的传输方案。

VII. 基于横向设计的编码缓存方案

考虑一个 TD(k, n) 横截设计 $(\mathcal{X}, \mathcal{G}, \mathcal{B})$，其中 $k \geq n$。我们以某种特定方式排列各组。因此， $\mathcal{G} = \{G_i: \ i \in \{0, 1, \ldots, k-1\}\}$ 其中 $|G_i| = n$ 表示各组。注意 $|\mathcal{X}| = kn$。由 (TD4) 可知，对于 $\lambda = 1$ ，来自 $\mathcal{X}$ 的每一对元素，要么恰好包含在 $\mathcal{G}$ 中的一个组内，要么包含在 $\mathcal{B}$ 中的一个区组内，但不会同时包含在两者中。因此，区组中的任意两个元素必定来自不同的组。于是，对于任意区组 $A \in \mathcal{B}$ $(|A| = k)$，恰有一个元素 $A(G_i)$，使得 $A(G_i) \in B \cap G_i$。因此，区组 $A \in \mathcal{B}$ 中的元素可以排列为 $\{A(G_0) \ldots, A(G_{k-1})\}$ 其中，元素 $A(G_i)$ $i \in \{0, 1, \ldots, k-1\}$ 来自组 G_i。

令 C 为 TD(k, n) 关联矩阵的转置。矩阵 C 的行数和列数分别为 n^2 和 kn 。

注3。 容易看出，矩阵 C 的每行和每列的权重分别等于 k 和 n 。因此，矩阵 C 是一个行列常重矩阵。

矩阵 C 将给出一个 $(n^2, kn, \frac{1}{n})$-缓存矩阵。根据 TD(k, n) 的 TD性质，回想一下，每个 $x \in \mathcal{X}$ 恰好出现在 $\mathcal{B}$ 的 n 个区组中。下面我们描述一个引理，该引理给出了矩阵 C 的一个单位子矩阵。

引理8。 令 $\mathcal{B}_x$ 表示包含 x 的区组集合，并记 $\mathcal{B}_x = \{A_1, \ldots, A_n\}$ ，其中 $x = A_i(G_{j_i}): \ j_i \in \{0, 1, \ldots, k-1\}$ ，即 x 是区组 A_i中的第 j_i 个元素。考虑子矩阵 C_x ，其行由$\mathcal{B}_x$ 索引，列由 $\{A_i(G_{j_i \oplus_k 1}): \ A_i \in \mathcal{B}_x\}$索引。则 C_x 是 C 的一个大小为 n的单位子矩阵。

证明：注意，C的子矩阵 C_x 中有 n 行。现在证明其中有 n 列。根据 $\lambda = 1$的TD性质，元素 $\{x, A_i(G_{j_i \oplus_k 1})\} \subset A_i$ 仅属于.....，且对任意$i' \neq i$，不属于任何其他 $A_{i'}$。因此，对任意 $i' \neq i$，元素 $A_i(G_{j_i \oplus_k 1}) \neq A_{i'}(G_{j_{i'} \oplus_k 1})$。所以，$C_x$中有 n 列。

任取一行，其索引为 $A_i \in \mathcal{B}_x$。假设该行在子矩阵 C_x 中索引为 y 和 y' 的两个不同列中，元素均为 1，其中 y，$y' \in \mathcal{X}$。这意味着 y，$y' \in A_i$，因而 $\{x, y, y'\} \subseteq A_i$，这意味着对于任意 $i \neq i'$，$\{x, y\} \nsubseteq A'_i$ 且 $\{x, y'\} \nsubseteq A'_i$。这意味着二者均有 y，$y' \in G_{j_i \oplus_k 1}$。根据 $\lambda = 1$ 的性质 (TD4)，这是不可能的。因此，子矩阵 C_x 的每一行中，元素 1 仅出现在一列中。

假设列 $A_i(G_{j_i \oplus_k 1})$ 中有两个元素为 1 的行，分别为 A_i 和 $A_{i'}$，且满足 $i \neq i'$。那么集合 $\{x, A_i(G_{j_i \oplus_k 1})\}$ 同时出现在区组 A_i 和 $A_{i'}$ 中。因此，这与 $\lambda = 1$ 的横截设计性质矛盾：元素对 $\{x, A_i(G_{j_i \oplus_k 1})\} \in \mathcal{X}$ 要么属于同一个组，要么恰好包含在一个区组中，但不能两者兼有。因此，C_x 是 C 中一个大小为 n 的单位子矩阵。

■

引理9。对于不同的 x_1、$x_2 \in \mathcal{X}$，不存在满足 $C(A, x) = 1$ 的 $x \in \mathcal{X}$、$A \in \mathcal{B}$，使得 $C(A, x)$ 同时被 C_{x_1} 和 C_{x_2} 覆盖。此外，矩阵集合 $\{C_x : x \in \mathcal{X}\}$ 构成 C 的一个大小为 n 的不重叠单位子矩阵覆盖。

证明：证明的第一部分采用与引理 3 相同的论证。矩阵 C 中 1 的总数等于每行中 1 的数量与行数的乘积，因此等于 kn^2。对于每个 $x \in \mathcal{X}$（注意 $|\mathcal{X}| = kn$），都存在一个单位子矩阵 C_x。由证明的第一部分和引理8可知，每个单位子矩阵的大小均为 n，且彼此不重叠。这些单位子矩阵将覆盖矩阵 C 中 kn^2 个 1，其数量等于 C 中 1 的总数。因此，$\{C_x : x \in \mathcal{X}\}$ 构成 C 的一个大小为 n 的不重叠单位子矩阵覆盖。

■

因此，我们得到以下定理来总结该缓存方案。

定理 4. *TD-(k, n)* 的关联矩阵的转置构成一个 $\left(K = n^2, F = kn, (1 - \frac{M}{N}) = \frac{1}{n}\right)$缓存矩阵。此外，存在一个速率为 $R = 1$ 的传输方案。

证明：缓存矩阵 C 的参数已定义。根据引理8和9，我们得到一个由 kn 个单位子矩阵组成的单位子矩阵覆盖。因此，根据定理 1，$S = kn$ 且速率为 $R = \frac{S}{F} = 1$。

■

例11. 考虑例*5*中给出的 *TD*(4,3) 。我们描述与 $\mathcal{X}$ 中元素‘2’对应的一些单位子矩阵 C_2 。元素‘2’出现在区组 $\{\{2, 4, 9, 11\}, \{2, 5, 7, 12\}, \{2, 6, 8, 10\}\}$ 中。因此，单位子矩阵矩阵 C_2 的行由 $\{2, 4, 9, 11\}, \{2, 5, 7, 12\}, \{2, 6, 8, 10\}$ 索引，列由4,5, 6 索引（因为4,5,6 分别是区组 $\{2, 4, 9, 11\}, \{2, 5, 7, 12\}, \{2, 6, 8, 10\}$ 中紧接在‘2’之后出现的元素），矩阵为 C。这样，子矩阵 C_i： $i \in \mathcal{X}$ 为我们提供了速率为1 的传输方案。

VIII. 基于 t-子空间设计的编码缓存方案

令 $(\mathcal{V}, \mathcal{A})$ 表示 $t \geq 2$ 的一个 t-$(v, k, 1)_q$-子空间设计。令 $\mathcal{A} = \{B_1, B_2, \ldots, B_b\}$ 表示该设计中的区组。令 T 表示 $\mathcal{V}$ 的所有 1维子空间构成的集合，H 表示所有 t 维子空间构成的集合，

$\mathcal{V}$，并令 R 表示 $\mathcal{V}$ 的所有 $(t-1)$ 维子空间构成的集合。

我们按如下方式构造二进制矩阵 C。令 C 的行由集合 R 编号。令列由 $\{(y, B): y \in T, y \subset B, B \in \mathcal{A}\}$ 编号。矩阵 C 的行数为 $\langle v, t-1\rangle$。矩阵 T 的列数为 $b\langle k, 1\rangle = \begin{matrix}\langle v, t\rangle\langle k\,1\rangle \\ \langle k\,t\rangle\end{matrix}$。对于某个 $D \in R$，矩阵 $C = (C(D, (y, B)))$ 按以下规则定义，

$$C(D,(y,B)) = \begin{cases} 1, & \text{if } D \bigoplus y \in H, D \bigoplus y \subset B \\ 0, & \text{otherwise.} \end{cases}$$

注4. T 的每一行中 1 的数量为 $\langle v-t+1, 1\rangle \cdot q^{t-1}$。注意，与某个固定的 s 维子空间在某个 l 维子空间中相交的不同 r 维子空间的数量为 $q^{(r-l)(s-l)}\langle v-s, r-l\rangle\langle s, l\rangle$（证明见 *[50]*），即可得出这一点。要计算每一行中1 的数量，由于每个 t 维子空间恰好包含在该设计的一个区组中，我们只需计算与一个固定的$(t-1)$ 维子空间（由该行标识）在一个 $(t-1)$ 维子空间中相交的 t 维子空间的数量。结果恰为 $\langle v-t+1, 1\rangle \cdot q^{t-1}$。同理，根据上述表达式可得每列中1 的数量为 $\langle k-1, t-1\rangle \cdot q^{t-1}$。因此，上述定义的二进制缓存矩阵 C 也是一个行列常重矩阵。

因此，我们看到矩阵 C 给出了一个 $\left(\langle v, t-1\rangle, \underbrace{\langle v, t\rangle\langle k\,1\rangle}_{\langle k\,t\rangle}, \ldots, \underbrace{\langle v-t+1, 1\rangle\langle k\,t\rangle q^{t-1}}_{\langle k\,1\rangle\langle v\,t\rangle}\right)$,,,-缓存矩阵。对于某个$y \in T$，令 $\mathcal{B}_y$ 为包含 y的区组集合。现在，

$$|\mathcal{B}_y| = \lambda_1 = \frac{\langle v-1, t-1\rangle}{\langle k-1, t-1\rangle}.$$

将 $\mathcal{B}_y$记为 $\mathcal{B}_y = \{B_1, B_2, \ldots, B_{\lambda_1}\}$。对于任意 B_i，用 $D_i = \{D_{i,j}: j \in \left[\langle k-1, t-1\rangle \cdot q^{t-1}\right]\}$ 表示 B_i 中所有不包含 y的（$t-1$）维子空间组成的集合。在下一个引理中，我们描述矩阵 C的一个单位子矩阵。

引理 10。对于某些 $j \in \left[\langle k-1, t-1\rangle \cdot q^{t-1}\right]$ 和 $y \in T$，考虑如下定义的 C 的子矩阵 $C_{y,\,j}$：$C_{y,\,j}$ 的行由 $\{D_{i,j}: \forall i \in [\lambda_1]\}$ 索引，列由 $\{(y, B_i): B_i \in \mathcal{B}_y\}$ 索引。那么，$C_{y,\,j}$ 是 C 的一个大小为 λ_1 的单位子矩阵。

证明：显然，$C_{y,\,j}$ 中行和列的数量均为 λ_1。因此，$C_{y,\,j}$ 是一个大小为 λ_1 的方阵。

现在，考虑 $C_{y,\,j}$ 中由 $D_{i,\,j}$ 索引的一行，其中 $i \in [\lambda_1]$. 为某个值。假设某个 $B \in \mathcal{B}_y$ 对应的列（y, B）在由 $D_{i,\,j}$索引的行中有一个 1。这意味着 $D_{i,\,j} \bigoplus y \subset B$；但是，根据矩阵 C 的定义规则，只有一个 B 满足 $D_{i,\,j} \bigoplus y \subset B$，而这恰好就是 $B = B_i$。因此，子矩阵 $C_{y,\,j}$ 中的每一行都只包含一个 1。

现在，考虑 $C_{y,j}$ 中由 (y, B_i) 索引的某一列。假设对于某一行 $D_{i',j}$，在由 (y, B_i) 索引的列中有一个 1。这意味着 $D_{i',j} \bigoplus y \subset B_i$，进而意味着 $D_{i,j} \bigoplus y \subset B_i \cap B_{i'}$。但是，根据具有 $\lambda = 1$ 的 t-子空间设计的性质，我们有 $i = i'$。因此，子矩阵 $C_{y,j}$ 中的每一列都只包含一个 1。这证明了该引理。

■

在接下来的两个引理中，我们将证明集合 $\{C_{y,j}: \forall y \in T,\ \forall j \in \left[\langle k-1, t-1\rangle \cdot q^{t-1}\right]\}$ 中的单位子矩阵彼此不重叠，并且这些单位子矩阵将覆盖矩阵 C 中所有满足 $C(D,(y,B)) = 1$ 的元素。

引理 11. 任何 $C(D,(y,B)) = 1$ 只要满足 $y \in T, B \in \mathcal{A}, D \in R$，就恰好会被 C 的一个单位子矩阵覆盖（定义见引理 *10*）。

证明：设 $C(D,(y,B)) = 1$ 被 C的一个单位子矩阵 $C_{y',j}$ 覆盖。由于 $C(D,(y,B)) = 1$，我们有 $D \bigoplus y \subset B$。根据引理 10 中 $C_{y',j}$ 的定义，首先必须有 $y = y'$。此外，对于某个包含 y的 B_i，必须满足 $D \subset B_i \backslash y$。因此，$D \bigoplus y \subset B_i$，这意味着 $B_i = B$（因为 $\lambda = 1$）。所以，覆盖 $C(D,(y,B)) = 1$ 的唯一传输是 $C_{y,j}$，其中 j 满足 $D = D_{i,j}$ 是 D_i中的唯一 j^{th} 集合。证明完毕。

■

引理 12. 矩阵集合 $\{C_{y,j}: \forall y \in T,\ \forall j \in \left[\langle k-1, t-1\rangle \cdot q^{t-1}\right]\}$ 构成了 C的不重叠单位子矩阵覆盖。

证明：C 中的 1 的总数等于每行中 1 的数量与行数的乘积，因此等于 $\langle v-t+1, 1\rangle\langle v, t-1\rangle q^{t-1}$。每个单位子矩阵的大小为 $\begin{matrix}\langle v-1\ t-1\rangle \\ \langle k-1\ t-1\rangle\end{matrix}$,,。由于对于每个 $j \in \left[\langle k-1, t-1\rangle q^{t-1}\right]$，$y \in T$，我们都有一个单位子矩阵 $C_{y,j}$，因此单位子矩阵的数量为 $\langle k-1, t-1\rangle\langle v, 1\rangle q^{t-1}$。此外，根据引理 11，各单位子矩阵之间没有重叠。因此，所有单位子矩阵覆盖的 1 的总数

$$\langle k-1, t-1\rangle\langle v, 1\rangle q^{t-1}\frac{\langle v-1, t-1\rangle v}{\langle k-1, t-1\rangle} = q^{t-1}\langle v-1, t-1\rangle\langle v, 1\rangle.$$

容易验证，上述数量等于矩阵 C 中 1 的总数。因此，$\{C_{y,j}: \forall y \in T,\ \forall j \in \left[\langle k-1, t-1\rangle \cdot q^{t-1}\right]\}$ 构成了 C 的不重叠单位子矩阵覆盖。

■

因此，我们得到以下定理来概述该缓存方案。

定理 5。 t-$(v,k,1)_q$-子空间设计的矩阵 C 构成一个 $\left(\langle v, t-1\rangle, \begin{matrix}\langle v,\ t\rangle\langle k\ 1\rangle, \ , \ , \ \langle v-t+1, \quad 1\rangle\langle k\ t\rangle q^{t-1}, \ , \ , \\ \langle k\ t\rangle \quad \cancel{\langle k\ 1\rangle\langle v} \\ \langle v\ t\rangle\cancel{\langle k\ 1\rangle}\end{matrix}\right)$-缓存矩阵。此外，存在一种速率为 $R = \begin{matrix}\langle k-1 \\ \ , \ t-1\rangle\langle v, \quad 1\rangle\langle k\ t\rangle q^{t-1}, \ , \ , \\ \langle v\ t\rangle\cancel{\langle k\ 1\rangle}\end{matrix}$ 的传输方案。

示例 12. 我们给出一个关于 $t = k$ 情形的示例，由此得到一个 k-$(v,k,1)_q$ 子空间设计。令 $q = 4$、$v = 3$ 和 $k = t = 2$。于是，我们得到一个参数为 $K = 21$、$1 - M/N = 0.19$、$F = 105$ 且速率为 $R = 0.8$ 的编码缓存方案。这也是第 *XIII* 节表 *IV* 中比较的数值示例之一。

IX. 编码 MapReduce 分布式计算的系统模型

在接下来的章节中，我们将讨论在 [10], [11] 中引入的编码 MapReduce 框架下，如何应用我们所构建的基于设计的二进制矩阵。为此，我们先简要回顾 [11]，中提出的正式系统模型：该模型的任务是使用 K 个服务器或计算节点（由集合 $\mathcal{K}$ 索引）处理一个大型文件，以计算 Q 个输出函数。我们记作 $\beta \triangleq \frac{Q}{K}$，并假设 β 是一个整数，如 [11] 中所述。该文件被划分为 F 个子文件（我们假设 $F \geq K$），并用大小为 F 的集合 $\mathcal{F}$ 表示（稍微滥用一下符号，

我们也将使用 $\mathcal{F}$ 为文件的 F 个子文件编索引）。参数 F 在编码分布式计算领域也称为文件复杂度。每个子文件分配给 r 个服务器，其中 r 称为计算负载。显然， $r \geq 1$。令 $\mathcal{M}_k \subseteq \mathcal{F}$ 表示分配给服务器 k 的子文件集合， $k \in \mathcal{K}$。

设待计算的输出函数记为 $\phi_1, \ldots, \phi_Q$，其中每个 ϕ_q 将所有输入文件映射到 u_q，且 $u_q = \phi_q($ $\{\forall f \in \mathcal{F}\})$ 是长度固定的二进制流。令 $g_{q,f}, \forall q \in [Q], \forall f \in \mathcal{F}$ 表示映射函数，它们将输入子文件 $f \in \mathcal{F}$ 映射为 Q 个中间值（IVA），每个中间值由 T 比特组成，记为 $\{v_{1,f}, v_{2,f} \ldots, v_{Q,f}\}$。每个 $v_{q,f} \triangleq g_{q,f}(f)$ $q \in [Q]$, $f \in \mathcal{F}$ 表示与 q^{th} 函数及子文件 f 对应的、长度为 T 比特的中间值。记为 $h_q, q \in [Q]$ 的归约函数将中间值 $v_{q,f}$: $\forall f \in \mathcal{F}$ 映射为输出比特流 u_q。因此，我们有 $u_q = \phi_q(\{\forall f \in \mathcal{F}\}) = h_q(\{v_{q,f}$: $\forall f \in \mathcal{F}\})$ $= h_q(\{g_{q,f}(f) : \forall f \in \mathcal{F}\})$。MapReduce 框架中的分布式计算方案包含三个阶段：映射、洗牌和归约阶段，下面将使用上述函数对其进行描述。

1) 映射阶段： 在映射阶段，每台服务器 $k \in \mathcal{K}$ 都会使用映射函数计算 $\mathcal{M}_k$ 中子文件的所有中间值，即服务器 k 计算 $g_{q,f}(f)$: $\forall f \in \mathcal{M}_k$、$\forall q$。因此，在映射阶段之后，服务器 $k \in \mathcal{K}$ 拥有$\{v_{q,f}$： $\forall q \in [Q]$、$\forall f \in \mathcal{M}_k\}$。

2) 洗牌阶段： 每台服务器负责对 Q 个函数中的一个不同的 $\beta = \frac{Q}{K}$ 函数子集执行归约（计算 h_q 函数）。令 $\mathcal{W}_k = \{q_{(k,b)}$： $\forall b \in [\beta]\} \subset [Q]$ 表示服务器 $k \in \mathcal{K}$上待归约函数的索引，其中 $\cup_{k=1}^{K} \mathcal{W}_k = [Q]$。为了计算一个归约函数的输出，服务器需要该输出函数针对所有子文件的中间值。除映射阶段中已针对 $\mathcal{M}_k$中的子文件计算出的中间值外，每台服务器 $k \in \mathcal{K}$ 还需要 $\{v_{q,f}$： $\forall q \in \mathcal{W}_k$、$\forall f \notin \mathcal{M}_k\}$，以归约分配给它的函数。因此，在洗牌阶段，各服务器相互发送广播传输，以确保每台服务器都能接收到执行分配给它的归约操作所需的中间值。

3) 归约阶段： 利用在洗牌阶段接收到的中间值以及在映射阶段本地计算出的中间值，服务器 k 使用归约函数 h_q 计算分配给它的任务，即，节点 k 对每个 $q \in \mathcal{W}_k$ 计算 $h_q(\{v_{q,f} : \forall f \in \mathcal{F}\})$，从而有效地计算出输入文件上函数 ϕ_q： $\forall q \in \mathcal{W}_k$ 的值。

与 [11] 中一样，分布式计算框架的归一化通信负载 L 定义为所有 K 服务器在洗牌阶段通信的比特总数（归一化后），其计算方法如下:

$$L \triangleq \frac{\text{Total number of bits transmitted in shuffle phase}}{QFT},$$

其中， T 是每个 IVA 的比特大小。在 [10], 中引入的编码分布式计算框架在洗牌阶段采用编码传输，以降低通信负载。研究表明，对于计算负载 r， [10], [11] 中的方案实现的通信负载为 $\frac{1}{r}(1 - \frac{r}{K})$，并且 [11] 证明该通信负载是最优的。

X. 二元矩阵与分布式计算

在第二节中，我们使用二进制矩阵来设计编码缓存方案。同样地，在本节中，我们将介绍如何从一个具有恒定列重的二进制矩阵导出分布式计算方案。

定义10（二元计算矩阵）。考虑一个二进制矩阵 C，其行由一个大小为 K的集合 $\mathcal{K}$ 索引，列由一个大小为 F的集合 $\mathcal{F}$ 索引，并且任意一列中0的数量恒定（设为 r）。则矩阵C 定义了一个分布式计算方案，其中包含 K 个用户（由 $\mathcal{K}$索引）、文件复杂度 F（子文件由 $\mathcal{F}$索引）和计算负载 $= r$，具体如下：

- 服务器 $k \in \mathcal{K}$ 映射子文件 f： $\forall f \in \mathcal{F}$，若 $C(k, f) = 0$；若 $C(k, f) = 1$，则不映射该子文件。

我们将矩阵 C 称为一个（K, F, r）-计算矩阵。

现在，我们给出一个计算矩阵的例子，并将在本节后文中用它来说明我们新提出的洗牌方案。

例13。考虑由 $\mathcal{K} = \{1, 2, 3, 4, 5, 6, 7\}$ 和 $\mathcal{F} = \{127, 145, 136, 467, 256, 357, 234\}$给出的集合系统（$\mathcal{K}, \mathcal{F}$）。该集合系统的关联矩阵 C 为

$$C = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 127 & 145 & 136 & 467 & 256 & 357 & 234 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \end{array}.$$

容易看出，这给出了一个(7,7,4)-计算矩阵。对应的分布式计算系统有$K = 7$ 个节点和 $F = 7$ 个子文件，其中每个子文件存储在 $r = 4$ 个用户中。例如，索引为127的子文件存储在用户3、4、5和6中。

为了描述进行编码传输的洗牌阶段，我们首先介绍基于计算矩阵的一个单位子矩阵进行的一轮两次传输，这些传输将服务于若干服务器。

引理 13。考虑 C的一个单位子矩阵，其行由 $\{k_1, k_2, .., k_l : k_i \in \mathcal{K}\}$ 给出，列由 $\{f_1, f_2, .., f_l : f_i \in \mathcal{F}\}$给出，满足 $C(k_i, f_i) = 1, \forall i \in [l]$，而 $C(k_i, f_j) = 0, \forall i, j \in [l]$，其中 $i \neq j$。那么，任意两个不同的服务器 k_i 和 k_j： $i, j \in [l], i \neq j$ 都可以进行两次传输，每次长度为 $\beta T = \frac{QT}{K}$ 比特，一次编码、一次未编码，使得每个服务器 k_i： $i \in [l]$ 都能从这两次编码传输中恢复缺失的 ***IVA***，即 $\{v_{q_{(k_i,b),f_i}} : \forall b \in [\beta]\}$。

证明： 根据单位子矩阵的定义，对于每个 $i \in [l]$，中间值 $\{v_{q_{(k_i,b),f_i}} : \forall b \in [\beta]\}$ 在服务器 k_i 上不可用，但在其他服务器 $\{k_1, k_2, \ldots, k_l\} \backslash k_i$ 上可用。因此，对于某个 $p \in [l]$，考虑编码

由服务器 k_p 进行的长度为 βT 的传输

$$\left\{\sum_{\substack{i=1\\ i\neq p}}^{l} v_{q_{(k_i,1)},f_i}, \sum_{\substack{i=1\\ i\neq p}}^{l} v_{q_{(k_i,2)},f_i}, \ldots, \sum_{\substack{i=1\\ i\neq p}}^{l} v_{q_{(k_i,\beta)},f_i}\right\}.$$

根据上述传输，每个 $i \in [l]\backslash p$ 显然都能恢复中间值 $\{v_{q_{(k_i,b)},f_i}: \forall b \in [\beta]\}$。例如，通过传输 $\sum_{i=1} v_{q_{(k_i,1)},\ f_i}$，服务器 k_j： $j \in [l]\backslash p$ 能够恢复中间值 $v_{q_{(k_j,1)},\ f_j}$，因为所有其他中间值 $_{i\neq p} v_{q_{(k_i,1)},\ f_i}$： $i \in [l]\backslash\{j,p\}$ 已在映射阶段存储于服务器 k_j。现在，任取一个服务器 k_i： $i \in [l],\ i \neq p$。令该服务器 k_i 按如下方式传输大小为 βT 比特的未编码传输。

$$\{v_{q_{(k_p,1)},f_p},\ v_{q_{(k_p,2)},f_p}, \ldots, v_{q_{(k_p,\beta)},f_p}\}.$$

因此，服务器 k_p 原样接收到中间值 $\{v_{q_{(k_p,b)},f_p}: \forall b \in [\beta]\}$。至此证明完成 . ■

A. 一种新的简单低复杂度编码数据洗牌算法

利用引理 13，我们给出以下定理，它为洗牌阶段提供了一种新的简易方案。

定理 6. 考虑一个大小为 $K \times F$ 的计算矩阵 C，其具有不重叠单位子矩阵覆盖 $\mathfrak{C} = \{C_1, C_2, .., C_S\}$，且每个单位子矩阵的大小为 $g \geq 2$。那么，存在一种包含 K 个节点的分布式计算方案，其计算负载为 r、通信负载为 $L = \frac{2}{g}(1 - \frac{r}{K})$，文件复杂度为 F。

证明：根据引理 13，对于 $\mathfrak{C}$ 中的每个单位子矩阵 C_i，都存在两次传输，在由 C_i 的行索引所标识的用户之间，交换与子矩阵 C_i 的列所对应子文件有关的所有缺失中间值（对应于各服务器上待归约的所有函数，这些服务器由 C_i 的行索引标识）。注意，由于 $\mathfrak{C}$ 是一个单位子矩阵覆盖，因此每个用户的每个缺失中间值都会包含在 $\mathfrak{C}$ 中某个单位子矩阵对应的某次传输中。因此，所有用户上与所有待归约函数对应的缺失中间值都能被解码。由此，各节点也能成功执行归约函数。由于 $\mathfrak{C}$ 包含 S 个单位子矩阵，传输总次数为 $2S$。每次传输的大小为 $\beta T = QT/K$ 比特。因此，通信负载为 $L = \frac{2QTS}{KQFT} = \frac{2S}{KF}$。每个单位子矩阵的大小为 g，因此每个单位子矩阵中的 1 的总数为 g。单位子矩阵之间互不重叠，因此计算矩阵中的 1 的总数为 Sg。此外，计算矩阵每列中的 1 的数量为 $K - r$，因为每个子文件存储在 r 个服务器上。因此，矩阵 C 中 1 的总数为，

$$Sg = F(K - r) \tag{4}$$

利用（4），我们有： $L = \frac{2(K-r)F}{KgF} = \frac{2}{g}(1 - \frac{r}{K})$. ■

现在我们给出一个示例来说明这一新方案，并接续例13，展示该例中计算矩阵的单位子矩阵覆盖。

示例 14（例13续）。例*13*中的矩阵里用7 种形状标出的单位子矩阵，显然构成了一个由7 个互不重叠的单位子矩阵组成的单位子矩阵覆盖。

$$C = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 127 & 145 & 136 & 467 & 256 & 357 & 234 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

现在展示一轮传输，其中包含两次传输，分别对应覆盖中的一个单位子矩阵。考虑其中一个单位子矩阵，记为 C_2，其中 C_2 如下所示。

$$C_2 = \begin{array}{c} \\ 3 \\ 5 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccc} 127 & 256 & 234 \end{array} \\ \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \end{array}$$

在 C_2中，$\{3,5,7\}$ 中的一个服务器将进行编码传输，另一个将进行未编码传输。设 $Q = 14$（即 $\beta = 2$）以及 $\mathcal{W}_3 = \{3,10\}$、$\mathcal{W}_5 = \{5,12\}$、$\mathcal{W}_7 = \{7,14\}$。对应于这个子矩阵，服务器 *3* 缺失的中间值为 $\{v_{3,234}, v_{10,234}\}$，即 $234 \notin \mathcal{M}_3$。类似地，服务器 *5* 和 *7* 缺失的中间值分别为$\{v_{5,256}, v_{12,256}\}$ 和 $\{v_{7,127}, v_{14,127}\}$。令服务器 *3* 进行编码传输，服务器 *5* 进行未编码传输。于是，服务器 *3* 的编码传输为 $\{v_{7,127} \oplus v_{5,256}, v_{14,127} \oplus v_{12,256}\}$。注意，服务器 *5* 已在映射阶段计算出$v_7$、*127* 和$v_{14}$、*127*，因此它可以从这次编码传输中解码出$v_5$、*256* 和$v_{12}$、*256*。类似地，服务器 *7* 也可以解码出所需的中间值。服务器 *3* 缺失的中间值可通过服务器 *5* 进行的未编码传输 $\{v_{3,234}, v_{10,234}\}$接收。以类似方式，对单位子矩阵覆盖中的每个此类单位子矩阵进行相应传输，各服务器均可成功解码所需的中间值，如定理 *6* 所述。注意，在这种情况下，每个单位子矩阵的大小为 *3*，因此速率为 *2 3*（$1 - \frac{4}{7}$） $= \frac{2}{7}$。

– – –

[11] 中的编码数据洗牌方案实现了 $L = \frac{1}{r}(1 - \frac{r}{K})$的通信负载，并且已证明这是最优的。现在我们回顾该方案，并使用计算矩阵介绍映射阶段，同时计算我们的新数据洗牌方案所实现的通信负载。以下推论给出了这一结果，该推论是定理 6 的推论。

推论 1。对于任意正整数 K和 $r \in [K]$，存在一个$(K, \binom{K}{r}, r)$计算矩阵，由此可得到一个在 K 个节点上运行的分布式计算方案，其计算负载为 r ，通信负载为 $L = \frac{2}{r+1}\left(1 - \frac{r}{K}\right)$，文件复杂度为$F = \binom{K}{r}$。此外，该负载 $L < 2L^*(r)$，其中 $L^*(r)$ 是给定计算负载 r时的最优速率。

证明：令 $\mathcal{K} = [K]$ 表示节点集合，$\mathcal{F} = \{f_A : A \in \binom{[K]}{r}\}$ 表示子文件集合，其中$\binom{[K]}{r}$ 表示 $[K]$. 的 r 元子集。考虑一个大小为 $K \times \binom{K}{r}$ 的矩阵 C，其元素为 $C(k, f_A)$，当 $k \in A$时，且

$C(k, f_A) = 1$，当 $k \notin A.$时。因此，矩阵 C是一个$(K, F = \binom{K}{r}, r)$-计算矩阵。很容易得到该矩阵的一个不重叠单位子矩阵覆盖，具体如下。考虑 $[K]$的一个大小为 $r+1$ 的子集，记为 B。容易验证，由 B 定义的行集合与列 $B\backslash\{k\}$ ： $k \in B$ 构成一个单位子矩阵，记为 C_B。此外，对于每个这样的 B，子矩阵 C_B 的大小为 $r+1$，并且很容易验证这些矩阵彼此不重叠。进一步，对于 $C(k, f_A) = 1$，其元素恰好由 $\{k\} \cup A.$定义的单位子矩阵覆盖。因此，单位子矩阵集合 $\{C_B\colon B \in \binom{[K]}{r+1}\}$是 C的一个不重叠单位子矩阵覆盖，其中 $g = r+1$。因此，根据定理 6，我们基于该计算矩阵 C的数据洗牌方案实现了通信负载$L = \frac{2}{r+1}(1 - \frac{r}{K})$。由于已知由 [11] 可得，计算负载为 r时的最优通信负载为 $L^*(r) = \frac{1}{r}(1 - r/K)$，比较这两个表达式可知，$L = \frac{2r}{r+1} L^*(r) < 2L^*(r)$。

—— — –

—— ■

1）与 *[11]* 中的最优方案相比，我们的洗牌方案具有以下优势： 推论 1 表明，与 [11] 中的最优方案相比，我们的方案具有更高的通信负载。现在，我们讨论一下与 [11] 中的洗牌方案相比，我们的方案所具有的优势。在 [11]，中的最优负载方案的数据洗牌阶段，中间值必须进一步划分为 r 个更小的分块，每个分块长度为 $\frac{T}{r}$ 比特。然后，对于 $[K]$ 中大小为 $r+1$ 的每个子集 B， B 中的每台服务器都会对一组 r 个分块进行编码，并将其广播给 B 中的其他服务器。我们的方案无需进一步划分中间值。将中间值进一步划分为 r 个更小的分块会带来多项开销，包括协调、索引、切换等方面的开销，下面我们将对此进行讨论。由于避免了对中间值进行分块，与 [11] 中的方案相比，我们将自己的方案称为低复杂度方案。

- 首先，为了识别各个分块，需要进行一些索引操作。这种分块索引开销是在原始文件复杂度 F 之外额外产生的。我们的新方案避免了进一步分块，因此不会产生这项开销。

- 因此，映射阶段中计算出相同 IVA 的多个服务器必须采用完全相同的 IVA 文件分块方式。否则将无法解码。因此，这种分布式 IVA 分块不同于原始的文件复杂度 F；后者在将数据放置到节点的存储中以进行映射之前完成，而且可能在单台机器上完成。因此，与我们的方案相比，这种针对 [11] 的分布式 IVA 分块需要进行更多协调，以确保各个节点之间达成一致。

- 此外，与执行次数较少但每次读取数据量较大的读取操作相比，从实际存储设备（例如硬盘或闪存盘）读取大量较小分块会消耗更多时间和电力（我们的读取操作会读取整个中间值，即方案 [11] 中读取大小的 r 倍）。

- 最后，假设每个节点上的传输按照其在不同的集合 $B \subseteq [K]$ 中出现的顺序依次进行，这些集合的大小均为 $r+1$。由于 B中的每个节点都会参与对应于 B的传输，因此需要额外付出代价，多次开启和关闭传输设备（次数为 $r\binom{K}{r}$） 。然而，在我们的方案中，每个 B对应的传输轮次仅有 2 台服务器参与。这意味着我们只需承担 $2\binom{K}{r}$ 次开关切换的代价。

B. 定理 *6* 中方案的通信负载均衡

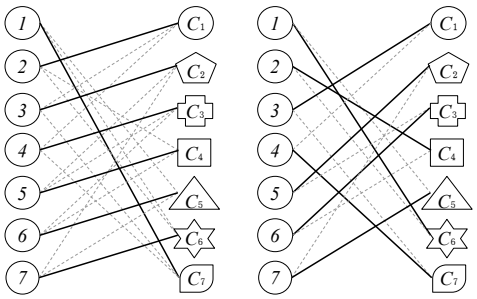
根据定理 6 的数据洗牌方案可确保，对于单位子矩阵覆盖 $\mathfrak{C}$中的每个单位子矩阵，只需由 2 台服务器进行传输。由于引理 13 任意选取这些服务器，因此可能导致通信负载不均衡：某些服务器传输的比特更多，而另一些服务器传输的比特少得多，甚至完全不传输。这种网络流量不均衡还可能引发其他问题，例如负载过重导致节点故障、整体性能下降等。下面的结果表明，通过在适当定义的二分图上识别两个完美匹配，可以解决负载不均衡问题（前提是满足一些简单条件）；而且，该操作可在参数 S的多项式时间内完成。这些条件适用于我们提出的某些构造，也适用于文献中的一些重要方案，例如 [11]中的方案。

定理7. 设 C是一个(K, F, r)-计算矩阵，具有一个不重叠单位覆盖 $\mathfrak{C} = \{C_1, \ldots, C_S\}$，其中每个单位子矩阵 C_i 的大小为 $g \geq 2$，表示一个可减少 Q 个函数的计算系统。那么，定理 *6* 中的编码数据洗牌方案可实现负载均衡，即每个节点传输的比特总数恰为 $\frac{2S\beta T}{K}$（其中 $\beta = Q/K$），其中 $\frac{S\beta T}{K}$ 比特为编码比特，其余 $\frac{S\beta T}{K}$ 比特为未编码比特，条件是：*(a)* $\gamma \triangleq \frac{S}{K}$ 为整数；*(b)* 若每个服务器 k出现在 $\mathfrak{C}$.中相同数量的单位子矩阵的行索引中

证明：我们首先构造一个二分图 $\mathcal{B}$。$\mathcal{B}$ 的左顶点集合是服务器集合（$\mathcal{K}$）重复γ 次，记为 $\{k^j : k \in \mathcal{K}, j \in [\gamma]\}$。右顶点分别是 $\mathfrak{C}$中单位子矩阵的索引。边定义如下。 k^j 与 C_i 之间存在一条边，当且仅当服务器 k 出现在单位子矩阵 C_i 所在的行中。由于 $\mathfrak{C}$ 中任意单位子矩阵的大小均为 g，因此图 $\mathcal{B}$ 是度为 $g\gamma$ 的右正则图；又由于左右顶点的基数相同（即 $\gamma K = S$），且满足性质 (b)，该图是度为 $g\gamma$ 的双正则图。

二分图上的完美匹配是一种匹配 M（由一组互不共享顶点的边组成），其中图中的每个顶点都至少与匹配中的一条边关联 M。对于具有 n 个顶点的正则二分图，可以在 $O(n \log n)$ [51] 时间内找到完美匹配。若在 $\mathcal{B}$ 中找到这样的完美匹配，则对于每个 $k \in \mathcal{K}$，集合 $\{k^j : j \in [\gamma]\}$ 中的每个左侧顶点都恰好与一个右侧顶点匹配，因此顶点 $\{k^j : j \in [\gamma]\}$ 与右侧顶点匹配（即 γ 单位子矩阵），例如 $\{C_{k_i} : i \in [\gamma]\}$。在这种情况下，我们指定顶点 k 负责与单位子矩阵$C_{k_i} : i \in [\gamma]$对应的编码传输。因此，每个顶点 k 负责 γ 次编码传输。

现在，为了定义未编码传输的服务器身份子矩阵配对，我们首先从 $\mathcal{B}$ 中移除与已找到的完美匹配相对应的一些边，从而得到一个度为 $(g\gamma - \gamma)$ 的新二部正则图。设对于某个 k, p 和 i，我们已找到的完美匹配中存在一条连接 k^p 和 C_i 的边，那么我们将从 $\mathcal{B}$ 中移除 k^j 和 C_i, $\forall j \in [\gamma]$ 之间的所有边。显然，$\mathcal{B}$ 的每个顶点的度都减少为 $g\gamma - \gamma = \gamma(g-1)$。注意，由于 $g \geq 2$，我们必有 $\gamma(g-1) \geq 1$。因此，我们得到一个度为 $\gamma(g-1)$ 的新正则图。于是，我们可以再次在该图上寻找完美匹配，例如使用 [51] 中的算法。类似于上一段中的论证，对于每个 $k \in \mathcal{K}$，我们



Fi图1：基于例13中计算矩阵的二分图，用于说明定理7

可以从新的完美匹配（记为 $\{C_{k_j}$ ）中得到与其相关联的另一组 γ 个身份： $j \in [\gamma]\}$。然而，由于上述边已被移除，我们必须有 $\{C_{k_i}$： $i \in [\gamma]\} \cap \{C_{k_j}$： $j \in [\gamma]\} = \phi$，即第一次匹配中与 k 相关联的身份在第二次匹配中均不会重复。因此，每个 $k \in \mathcal{K}$ 都可以被分配 γ 个与 γ 个身份子矩阵相对应的未编码传输；这些身份子矩阵均不同于 γ 已被分配用于执行编码传输的身份。现在，对于每个身份子矩阵（右侧顶点），我们得到了两条分别由上述方式获得的两个完美匹配产生的边，而这两条边必然分别关联于两个节点 $k_1^{j_1}$，$k_2^{j_2}$，其中 $k_1 \neq k_2$。因此，对于每个身份子矩阵，我们从第一次和第二次完美匹配中分别确定了两个不同的服务器 k_1 和 k_2 ，它们分别执行编码传输和未编码传输。结合对负责编码传输和未编码传输的服务器节点的确定，以及定理 6 中继续保持不变的论证，证明完毕。

■

例15。 考虑例*13*中给出的单位子矩阵，设想一个分布式计算场景，其中 $K = 7$ 和 $F = 7$对应的单位子矩阵覆盖由例*13*中的 $S = 7$ 个矩阵给出。因此， $\gamma = \frac{S}{K} = 1.$ 为简单起见，假设需要约简 $K = Q$ 个函数，即 $\underline{\beta} = 1$。图*1*左侧的第一幅图展示了按照定理*7*构造的二分图，左侧为用户顶点，右侧为使用不同形状表示的*7*个单位子矩阵。第一幅图中的粗边表示所获得的第一个完美匹配，利用该匹配分配编码传输。例如，用户 6 参与了与例*13*中对应于“△”形状的单位子矩阵相关的编码传输。删除这些边后，我们得到右侧的二分图，它同样是一个正则图。该图中的粗边表示与未编码传输对应的完美匹配。因此，在这种情况下，每个用户均参与 2 次传输，因为 $\beta = 1$，并且 $S = K = 7$，总共传输 $2T$ 比特，正如定理*7*所述。

C. 基于组合设计中的二进制矩阵的低文件复杂度（F）方案 s

在第五至第八节中，我们使用组合的设计和子空间设计构造了恒定行重的二进制矩阵，并将其用于编码缓存。在备注 1–4 中可以看到，这些二进制矩阵还具有

组合的 设计 ($\lambda=1$)	数量 服务器 K	File 复杂度 F	计算负载 r	通信 负载 非完全/部分掉队者情况	通信 负载 $K-\kappa$ 完全掉队者情况
BIBD	v	$\frac{v(v-1)}{k(k-1)}$	$(v-k)$	$\frac{2k(k-1)}{v(v-1)}$	$\frac{2k(k-1)}{\kappa(v-1)}$
t-设计	$\binom{v}{t-1}$	$\frac{\binom{v}{t}k}{\binom{k}{t}}$	$\binom{v}{t-1}-\binom{k-1}{t-1}$	$\frac{2(v-t+1)\binom{k-1}{t-1}^2}{v\binom{v-1}{t-1}^2}$	$\frac{2\binom{k-1}{t-1}^2}{\kappa\binom{v-1}{t-1}}$
横截 设计	n^2	kn	$n(n-1)$	$\frac{2}{n^2}$	$\frac{2}{\kappa}$
子空间设计	$\langle v,t-1\rangle$	$\frac{\langle v,\ t\rangle\langle k,\ 1\rangle}{\langle k,\ t\rangle}$	$\langle v,\ t-1\rangle-\langle k-1,\ t-1\rangle q_{t-1}$	$\frac{2\langle k-1,\ t-1\rangle^2 q^{t-1}}{\langle v,\ t-1\rangle\langle v-1,\ t-1\rangle}$	$\frac{2\langle k-1,\ t-1\rangle^2 q^{t-1}}{\kappa\langle v-1,\ t-1\rangle}$

表 II：基于第五至第八节中构造的组合设计和子空间设计的分布式计算方案的参数。注意，参数 v，k，t，n 分别对应相应设计的参数。

恒定列重。因此，这些矩阵也可用作计算矩阵。表 II 展示了基于这些设计构造的计算矩阵所得到的分布式计算方案的参数。请注意，虽然表 II（第 2、3 列）中的 K 和 F 分别对应于第五至第八节所设计矩阵的行数和列数，但计算负载 r 是这些矩阵的行数与列重（由备注 1–4 得出）之差。此外，表 II 第 5 列中的通信负载是根据定理 6 计算的，并依据引理 2、5、8 和 10 得出的各方案覆盖中单位子矩阵的大小确定。表 II 的最后两列涉及掉队者场景，我们将在第十一节中讨论。第 XIII 节中，我们将这些方案与 [11] 中的基准方案进行比较（在零掉队节点的情况下，该方案对应于 [17] 中的方案）。最后，根据定理 7，要实现负载均衡，单位子矩阵的数量 S 必须能被 K（服务器数量）整除。这一条件仅对基于平衡不完全区组设计和横截设计的方案成立。基于 t 设计的方案无法实现负载均衡，因为对该方案而言，通常 K 不能整除 S。基于子空间设计的方案也有同样的问题。

XI. 对拖尾者场景的扩展

分布式计算框架面临的实际挑战之一是存在掉队节点。掉队节点是指执行操作速度比其他节点慢的节点。本节利用我们的方案（定理 6、引理 13）的优势：每个传输轮次（对应一个单位子矩阵）只涉及两台服务器进行通信。我们利用这一优势，在掉队者数量不超过固定值时实现掉队者鲁棒性，即 $g-2$，其中 g 是任意单位矩阵的大小。在第十一节 A 中考虑的完全掉队场景下，掉队节点的速度慢到无法完成分配给它们的任何映射任务，即可将其视为故障节点。因此，它们不参与映射、洗牌或归约阶段。我们进行比较的方案 [17], 也采用了类似的设置。在第十一节 B 中考虑的部分掉队场景下，掉队节点的速度比其他节点慢，减速因子为某个值，但它们并非故障节点。

在掉队者场景下，洗牌阶段的目标保持不变：在节点之间交换消息，使各节点能够获得归约阶段所需的中间值。因此，我们将掉队者场景下的通信负载重新定义为

$$L(\kappa) = \frac{\text{Number of bits transmitted in shuffle phase in 'worst case'}}{QFT}$$

其中，“最坏情况”指的是 $K-\kappa$ 个（$or\ K-\kappa'$）掉队者中最糟糕的子集（即导致负载最大的那个子集）。

A. 完全掉队者

本节将讨论如何使用计算矩阵来应对完全掉队者。这里提出的方案能够应对多达 $g-2$ 个掉队者。我们考虑这样一种设置：一旦 κ，$\kappa \in \mathcal{K}$ 个服务器完成映射操作，数据洗牌阶段便可开始。我们将这 κ 个服务器称为存活服务器，其余的$K-\kappa \in [0:\ g-2]$ 个服务器则为完全掉队者。为简单起见，我们假设 $\frac{Q}{\kappa}$ 是一个整数，因此 Q 个函数在 κ 个存活服务器之间均匀分配。下面描述一种能够应对 $g-2$ 个完全掉队者的方案。

定理 8。考虑一个大小为 $K \times F$ 的计算矩阵，其具有不重叠单位子矩阵覆盖 $\mathfrak{C} = \{C_1, C_2, .., C_S\}$，且每个单位子矩阵的大小为 $g \geq 2$。那么，存在一种具有 K 个节点、能够应对 $K-\kappa \in [0:\ g-2]$ 个完全掉队者的分布式计算方案，其计算负载为 $= r$，通信负载为 $L(\kappa) = \frac{2}{g}\left(\frac{K}{\kappa} - \frac{r}{\kappa}\right)$，文件复杂度为 F。

证明：假设任意一组 $K-\kappa$ 节点发生故障。由于 $K-\kappa \leq g-2$，对于覆盖 $\mathfrak{C}$. 中的任意单位子矩阵 C_i 的行而言，至少仍有 2 台服务器存活。因此，每个单位子矩阵都可使用两台服务器进行引理 13 中所述的传输。因此，可以采用与定理 6 类似的方案，不同之处在于，每台服务器被分配来减少 $\frac{Q}{\kappa}$ 函数，而非像定理 6 中那样减少 $\frac{Q}{K}$。按照定理 6 中类似的论证，本例中的通信负载计算如下。

$$L = \frac{2QST}{\kappa QFT} = \frac{2S}{\kappa F}$$

利用 (4)，可得：

$$L = \frac{2(K-r)F}{\kappa g F} = \frac{2}{g}\left(\frac{K}{\kappa} - \frac{r}{\kappa}\right).$$

注意到我们考虑的是任意一组掉队者，证明完毕。 ■

注 5. 在 *[17]*（以及扩展版本 *[34]*）中，作者采用 *[11]* 的方案（如推论 *1* 所示），以抵御任意集合的 $K-\kappa$ 个掉队者，其中 $K-\kappa \leq r-1$，并实现以下表达式给出的负载

$$L^*(\kappa) = \left(1 - \frac{r}{K}\right) \sum_{i=r+\kappa-K}^{min\{r,\kappa-1\}} \frac{1}{i} \frac{\binom{r}{i}\binom{K-r-1}{\kappa-i-1}}{\binom{K-1}{\kappa-1}}.$$

分布式计算参数 MAN-PDA	非 掉队者数量 κ	最优通信负载 [17] 中	通信负载中的 定理 8
$K=5, r=2, F=10, g=3$	5	0.3	0.4
$K=5, r=2, F=10, g=3$	4	0.45	0.5
$K=7, r=4, F=35, g=5$	7	0.107	0.171
$K=7, r=4, F=35, g=5$	6	0.13	0.2
$K=7, r=4, F=35, g=5$	5	0.17	0.24
$K=10, r=3, F=120, g=4$	10	0.23	0.35
$K=10, r=3, F=120, g=4$	9	0.27	0.39
$K=10, r=3, F=120, g=4$	8	0.3305	0.4375

表 III：对 [17] 中方案与定理 8 的通信负载进行数值比较，其中计算矩阵源自 [11] 中的方案（注 5）。

事实上，在给定参数 K, r 和掉队者数量 $K-\kappa \leq r-1$ 的情况下，该负载恰好是最优的（见 *[34]*）。根据推论 *1* 并利用定理 *8*，我们可以看到，对于 *[11]* $(K, F=\binom{K}{r}, r)$ 计算矩阵（其中 $g=r+1$），采用相同的方案，可以得到一种能够抵抗 $K-\kappa$ 个掉队者的分布式计算方案（其中 $K-\kappa \leq g-2=r-1$），其负载为 $L=\frac{2}{(r+1)}(K-r)$。我们将 K, r, κ 的不同取值代入 *[17]* 的方案（已证明该方案是最优的）以及我们的方案中，并在表 *III* 中比较完全掉队者情况下的通信负载。我们注意到，虽然我们的方案负载较高，但仍具有可比性。注意，对于我们的方案和 *[17]* 中的方案，文件复杂度均为 $F=\binom{K}{r}$。然而，我们的方案仍保留了第 *X-A1* 节中提到的优势。

我们还在表 II（最后一列）中列出了基于定理 8 获得的设计型计算方案的负载。请注意，根据定理 8 和定理 6 可知，在 $(K-\kappa)$ 个节点全部为掉队节点的情况下，负载是非掉队者负载的 $\frac{K}{\kappa}$ 倍。第 XIII 节给出了将这些方案的通信负载与 [17] 中方案进行比较的数值模拟结果。

B. 部分掉队者

本节使用计算矩阵描述一种存在部分掉队者的分布式计算方案。部分掉队者是指执行分配给自己的任务时比其他服务器更慢、但并未发生故障的服务器。与其他非掉队服务器相比，部分掉队者需要更多时间才能完成映射阶段。我们考虑这样一种场景：共有 K 个服务器，其中 κ' 个为非掉队者，$K-\kappa' \in [0$ 个为部分掉队者。我们给出一种方案，在该方案中，部分掉队者在洗牌阶段无需传输任何内容，但仍负责对分配给它们的输出函数进行归约。针对类似的部分掉队者场景，[18] 提出了一种分布式计算方案。

现在简要介绍 [18] 中讨论的方案。[18]，中的方案采用一种称为 g-正则 PDA 的组合结构，它与计算矩阵非常相似，其中每个单位子矩阵的大小为 g。[18] 中的方案计算 $Q=K$ 个输出函数（即 $\beta=1$），并且能够处理 $K-\kappa'<g-1$ 个部分掉队者。此外，在映射阶段，[18]，中包括部分掉队者在内的所有服务器，都会为存储在其中的文件计算所有中间值

也就是说，服务器 $k \in \mathcal{K}$ 会在映射阶段计算 $\{v_{q_{(k,1)},f}: \forall q_{(k,1)} \in [Q], \forall f \in \mathcal{M}_k\}$。[18] 中方案的通信负载为 $L = {}_{g-1-}{}^{1}{}_{(K-\kappa')}\left(1 - {}^{r}_{K}\right)$。

在本工作中，我们提出了一种基于计算矩阵的新方案，可处理部分掉队者。该方案可以计算 $Q = \beta K$ 个输出函数（其中 β 是整数），同时要求部分掉队者计算的中间值数量少于 [18] 中的数量。为简单起见，我们假设 Q 个归约函数均匀分配给 K 台服务器。我们将映射阶段分为三个子阶段。在我们的方案中，部分掉队者只参与映射子阶段 1 和 2 以及归约阶段，而不参与映射子阶段 3 或洗牌阶段。

设 C 是一个大小为 $K \times F$ 的计算矩阵，其不重叠单位子矩阵覆盖为 $\mathfrak{C} = \{C_1, C_2, .., C_S\}$，且每个单位子矩阵的大小为 $g \geq 2$。设 $\mathcal{I}_k$ 为包含服务器 $k \in \mathcal{K}$ 的行索引的单位子矩阵集合。由于计算矩阵 C 的 k^{th} 行（对任意 $k \in \mathcal{K}$）中的每个 1 都对应一个不同的单位子矩阵，因此，$|\mathcal{I}_k|$ 等于计算矩阵 C 的 k^{th} 行中 1 的数量。对于大小为 g 的单位子矩阵 $C' \in \mathfrak{C}$，我们将 $\mathfrak{I}' = \{(k^1, f^1).....,(k^g, f^g)\}$ 记为 $\mathcal{K} \times \mathcal{F}$ 中在子矩阵 C' 内取值为 1 的索引集合。

我们将映射阶段分为三个子阶段。算法 1 给出了各服务器在不同 Map 子阶段中计算 IVA 的详细过程。服务器按如下顺序 计算 IVA:

- Map 子阶段 1：服务器只计算其负责归约的函数所需的 IVA。
- Map 子阶段 2：服务器将计算解码其参与的计算矩阵中单位子矩阵所对应的编码传输所需的 IVA（针对其无需归约的函数）。
- Map 子阶段 3：最先完成 Map 子阶段 1 和 2 的前 κ' 个服务器进入 Map 子阶段 3。我们将这组 κ' 个服务器称为活动集合 $\mathfrak{D}$。活动集合 $\mathfrak{D}$ 一旦形成，其他不在 $\mathfrak{D}$ 中的 $K - \kappa'$ 个服务器在完成 Map 子阶段 2 后便停止映射操作。在 Map 子阶段 3 中，活动集合 $\mathfrak{D}$ 中的每个服务器都会为其所有已存储、且尚未在 Map 子阶段 1 和 2 中计算过的子文件计算 IVA。

算法 1 以更正式的方式给出了这一过程。注意，参与 C' 的每台服务器都会计算 $(g-1)\beta$ 个中间值。由于每台服务器都出现在 $|\mathcal{I}_k|$ 个单位子矩阵中，因此部分掉队服务器仅计算 $(g-1)\beta|\mathcal{I}_k| + \beta|\mathcal{M}_k|$ 个中间值；当 $\beta > 1$ 时，该数量严格小于 $Q|\mathcal{M}_k|$。按照算法 1 完成映射阶段后，仅由活动集合中的服务器通过编码传输交换中间值。以下定理描述了计算方案的数据洗牌阶段，该方案的映射阶段如算法 1 所述，并且能够容忍多达 $g-2$ 个部分掉队者，其中 g 是计算矩阵的一个参数。

定理 9。考虑一个大小为 $K \times F$ 的计算矩阵，其具有不重叠单位子矩阵覆盖 $\mathfrak{C} = \{C_1, C_2, .., C_S\}$，且每个单位子矩阵的大小为 $g \geq 2$。那么，存在一种含有 K 个节点的分布式计算方案，该方案能够容忍 $K - \kappa' \in [0: \ g-2]$ 个部分掉队者，实现计算负载 $= r$ 和通信负载 $L(\kappa') = \frac{2}{g}\left(1 - \frac{r}{K}\right)$，文件复杂度为 F。

算法 1： 计算 Map 子阶段中的 IVA

Map 子阶段 *1*：

并行执行

对每个 $k \in \mathcal{K}$，计算 $v_{q_{(k,b)},\ f}$： $\forall b \in [\beta],\ \forall f \in \mathcal{M}_k.$

end

Map 子阶段 *2*：

并行执行

对每个 $k \in \mathcal{K}$

对于 $C' \in \mathcal{I}_k$ **执行**

计算 $v_{q_{(k',b)},\ f'}$： $\forall b \in [\beta],\ \forall\ (k',\ f')\ \in \mathfrak{I}' \backslash\ (k,\ f)$。

end

end

令 $\mathfrak{D}$ 表示最先完成子阶段 1 和 2 的 κ' 台服务器。

映射子阶段 *3*：

并行执行

对于每个 $k \in \mathfrak{D}$，计算 $v_{q_{(k,b)},\ f}$： $\forall q_{(k,b)} \in [Q],\ \forall f \in \mathcal{M}_k$，如果在子阶段 1 中尚未计算，且子阶段 2。

end

证明： 假设任意 $K - \kappa'$ 个节点掉队。由于 $K - \kappa' \leq g - 2,$ 对于覆盖 $\mathfrak{C}.$ 中任意单位子矩阵的行，至少仍有 2 个活动服务器存活 C_i 因此，每个单位子矩阵都至少有两个活动服务器可用于执行引理 13 中所述的传输。

现在，在算法 1 的 Map 子阶段 2 中，服务器（包括掉队者和非掉队者）已经计算出其为解码编码传输所需的中间值。因此，每个单位子矩阵中的服务器都能够解码编码传输，从而获得其执行归约阶段中的归约操作所需的缺失（未映射）中间值。因此，类似于定理 6 中的方案是可行的，并且可以沿用定理 6 中的类似论证来推导通信负载。

■

十二、讨论

我们提出了利用源自组合设计及其 q类比的二进制矩阵构造的新型编码缓存方案和分布式计算方案。我们的方案主要适用于本地缓存较大的场景。采用更广泛的设计类别（尤其是相应设计的 λ 参数值更高的设计）可能会降低所需的缓存大小。然而，为这些方案设计传输方案（单位子矩阵覆盖）似乎较为困难。我们还针对一些基准方案给出了数值结果，以展示我们方案的优势。在本工作的早期版本之前， [1], [2], 该工作 [27] 以不同方式使用称为可分解设计的特殊组合结构来构造编码缓存方案。在 [1], [2], 出现之后，其他一些工作（例如 [28]）也进一步发展了我们的工作。称为交叉可分解设计的特殊可分解设计也被用于多接入编码缓存场景（其中每个客户端

可以连接到多个缓存）（例如，参见 [52]）。我们相信，缓存辅助通信这一场景为进一步深入应用组合设计和子空间设计理论提供了肥沃的土壤。

对于基于 MapReduce 的分布式计算，与现有的最优方案相比，我们提出了一种新的简单编码洗牌方案，该方案以通信负载略有增加为代价，避免了进一步进行 IVA 分块。我们还给出了新方案实现服务器间通信负载均衡的条件。我们也将所提出的方案扩展到了完整和部分掉队者场景。与在类似的 K, r 取值下的最优方案相比，利用由某些组合设计和子空间设计生成的二进制矩阵所构造的分布式计算方案，具有文件复杂度很低的优点，但代价是速率较高。

XIII. 数值比较

本节将对我们的方案与一些现有的重要基准方案进行数值比较。尽管自这些基准方案问世以来，文献中又提出了许多其他方案，但我们主要选择这些方案进行比较，因为它们具有传输率最优性，或者在实现接近最优速率时对子分组化的要求较低。基于这些原因，这些方案在性能方面仍然十分重要且具有竞争力。

关于编码缓存，我们给出表 IV 和表 V。表 IV 展示了表 I 中编码缓存方案的一些数值示例。通过使客户端数量 K 和未缓存比例 $1-\frac{M}{N}$ 相同，我们从子分包级别和速率两个方面，将这些方案与 [3]中的最优速率方案进行比较。在大多数情况下，我们发现，我们的方案的子分包级别 F 低于 [3]的子分包级别 $F^* = \binom{K}{MK/N}$（某些情况下差距相当大），而我们的方案的速率 R 通常高于 $R^* = K(1-M/N)/(1+MK/N)$中方案的速率（不过增幅不到一个数量级）。在表 V 中，我们将我们的方案与 [21]中的方案进行数值比较。 [21] 中的方案是编码缓存设置 [3],下已知的最佳方案之一，因为一般而言，它能提供比 [3] 更低的子分包级别，且速率不会大幅增加。 [21] 中的方案依赖于相关二分图的强边着色；该二分图使用集合论方法构造，涉及四个非负整数参数 m、a、b、λ，其中 a、$b \le m$ 和$\lambda \le \min(a$、$b)$。我们将 [21] 中方案的编码缓存参数记为 K_{SEC}、$(1-\frac{M}{N})_{SEC}$、R_{SEC}和 F_{SEC}，其表达式见表 V 的标题说明。在这种情况下，要将我们的参数与 [21]精确匹配有些困难。我们尽可能匹配客户端数量，并比较其余参数。我们发现，在某些情况下，我们的方案更具优势，而在另一些情况下， [21] 中的方案则占优。

在表 VI 中，我们给出了一些基于设计的构造用于编码计算方案的数值示例。我们给出所用设计的参数以及由此得到的编码计算方案参数，包括服务器数量 K、文件复杂度 F、计算负载 r，以及定理 6 给出的无掉队者场景下的通信负载（该值与定理 9 给出的部分掉队负载一致），还有在 $K-\kappa \in \{1,2\}$ 个掉队者情况下的完全掉队负载（由定理 8 给出）。最后，在表 VII 中，我们将我们的方案与 [17] 中的方案进行比较。可以看到，我们的方案在文件复杂度方面具有优势，但通信负载有所增加。

组合设计 ($\lambda=1$)	参数（来自特定构造设计）	$1-\frac{M}{N}$	K	R	F	R^* 来自 [3]	F^* 来自 [3]
对称平衡不完全区组设计，第 IV-A1 节	$n=5$	0.193	31	1	31	0.23	7.36×10^5
BIBD，第 IV-A1 节	$n=5$	0.2	25	0.83	30	0.24	5.3×10^4
t-设计 第 IV-A2 节	$q=2$	0.1	10	0.16	15	0.1	10
横截设计，第 IV-A3 节	$q=5$	0.2	25	1	25	0.24	5.3×10^4
子空间设计，第 IV-A4 节	$2-(3,\ 2,\ 1)_2$	0.28	7	0.67	21	0.324	21
子空间设计，第 IV-A4 节	$2-(4,2,1)_2$	0.133	15	0.28	105	0.142	105
子空间设计，第 IV-A4 节	$2-(3,\ 2,\ 1)_3$	0.23	13	0.75	52	0.27	286
子空间设计，第 IV-A4 节	$2-(3,\ 2,\ 1)_4$	0.19	21	0.8	105	0.22	5985
子空间设计，第 IV-A4 节	$2-(3,\ 2,\ 1)_5$	0.16	31	0.83	186	0.183	169911
子空间设计，第 IV-A4 节	$2-(3,\ 2,\ 1)_7$	0.122	57	0.875	456	0.136	2.6×10^8

表 IV：基于本文特定构造的编码缓存方案与 [3]的下界的数值比较。请注意，这些方案确实存在，因为已知这些设计存在。

组合的设计 ($\lambda=1$)	参数（来自特定构造的设计和SEC方案）	$1-\frac{M}{N}$	K	R	F	$(1-\frac{M}{N})_{SEC}$	K_{SEC}	R_{SEC}	F_{SEC}
BIBD	$n=4,\ (m,\ a,\ b,\ \lambda)=(6,\ 2,\ 3,\ 0)$	0.25	16	0.8	20	0.2	15	0.3	20
BIBD	$n=7,\ (m,\ a,\ b,\ \lambda)=(8,\ 4,\ 3,\ 3)$	0.143	49	0.875	56	0.071	70	0.143	56
对称平衡不完全区组设计	$n=4, (m,a,b,\lambda)=(7,2,2,0)$	0.238	21	1	21	0.476	21	1.667	21
对称平衡不完全区组设计	$n=7, (m,a,b,\lambda)=(8,3,3,3)$	0.14	57	1	57	0.018	56	0.018	56
横截设计	$q=9,\ (m,\ a,\ b,\ \lambda)=(9,\ 3,\ 3,\ 2)$	0.111	81	1	81	0.214	84	0.857	84

表 V：基于组合设计的特定构造（第 IV 节）的编码缓存方案与强边染色（SEC）方案 [21] 的数值比较。注意，$(1-\frac{M}{N})_{SEC}=\binom{a}{\lambda}\binom{m-a}{b-\lambda}\binom{m}{b}$，$K_{SEC}=\binom{m}{a}$，$R_{SEC}=\binom{m}{a}\binom{a}{\lambda}\binom{m-a}{b-\lambda}\binom{m}{b}$ 最大值$\{\binom{a+b-2\lambda}{a-\lambda},\binom{m-a-b+2\lambda}{\lambda}\}$，$F_{SEC}=\binom{m}{b}$。

组合的 设计 ($\lambda=1$)	参数 （来自特定 构造 的设计）	数量 服务器） K	File 复杂度 F	计算 Load r	L 对于 非/部分 掉队者情况	L 对于 $K-\kappa=1$	L 用于 $K-\kappa=2$
BIBD	$n=7$	49	56	42	0.0357	0.0364	0.0372
对称平衡不完全区组设计	$n=5$	31	186	26	0.0645	0.0667	0.0689
t-设计	$q=3$	45	120	42	0.0111	0.0113	0.0116
横截设计	$q=5$	25	25	20	0.08	0.083	0.087
子空间设计	$2-(3,\ 2,\ 1)_2$	7	21	5	0.19	0.22	0.27
子空间设计	$4-(5,4,1)_2$	155	465	147	0.00688	0.00693	0.00697
子空间设计	$3-(4,3,1)_3$	130	520	121	0.01065	0.01073	0.01082
子空间设计	$4-(5,4,1)_3$	1210	4840	1183	0.0011157	0.0011166	0.0011175

表 VI：我们的方案在非/部分掉队者情况和掉队者情况下的通信负载数值比较（将表 II 中的表达式应用于第 IV 节的具体构造）。

组合的 设计 ($\lambda=1$)	数量 服务器 K	计算 Load r	File 复杂度 F	File 复杂度 F 中 [17]	数量 非掉队者 κ	通信 负载为 定理 8	最优 通信 [17]中的负载
BIBD	25	20	30	53130	25	0.067	0.01
BIBD	25	20	30	53130	23	0.072	0.011
BIBD	121	110	132	1.28×10^{15}	119	0.0154	0.0008
对称平衡不完全区组设计	31	26	186	169911	31	0.065	0.006
对称平衡不完全区组设计	31	26	186	169911	29	0.069	0.0067
对称平衡不完全区组设计	133	122	1596	3.78×10^{15}	131	0.01526	0.00069
t-设计	45	42	120	14190	45	0.0111	0.0016
t-设计	45	42	120	14190	43	0.01163	0.00167
t-设计	1225	1204	2800	1.17×10^{45}	1223	0.0006	1.42616×10^{-5}
横截设计	25	20	25	53130	25	0.08	0.01
横截设计	25	20	25	53130	23	0.087	0.011
横截设计	49	42	49	8.6×10^{7}	47	0.04255	0.00354
子空间设计	13	10	52	286	13	0.115	0.023
子空间设计	13	10	52	286	11	0.136	0.028
子空间设计	130	121	520	2.2×10^{13}	130	0.0106	0.00057
子空间设计	130	121	520	2.2×10^{13}	128	0.0108	0.00058
子空间设计	1210	1183	4840	1.18×10^{55}	1210	0.001115	1.887×10^{-5}
子空间设计	1210	1183	4840	1.18×10^{55}	1208	0.001117	1.889×10^{-5}

表 VII：本文提出的方案与基于设计的计算方案之间的数值比较。 [17] 中方案的负载表达式见注 5，而本文方案的负载表达式汇总于表 II（结合第 IV 节中的具体构造）。

参考文献

[1] S. Agrawal、K. V. Sushena Sree 和 P. Krishnan，《基于组合设计的编码缓存》，载于 2019年 IEEE信息论国际研讨会（ISIT），第 1227–1231 页，2019年。[2] S. Agrawal 和 P. Krishnan，《通过二进制矩阵实现低复杂度分布式计算及其在掉队者场景下的扩展》，载于 2020年 IEEE信息论国际研讨会（ISIT），第 162–167 页，2020年。[3] M. A. Maddah-Ali 和 U. Niesen，《缓存的基本极限》，IEEE信息论汇刊，第 60 卷，第 5 期，第 2856–2867 页，2014年。[4] K. Wan、D. Tuninetti 和 P. Piantanida，《一种采用未编码缓存放置的缓存索引编码方法》，IEEE信息论汇刊，第 66 卷，第 3 期，第 1318–1332 页，2020年。[5] M. Ji、G. Caire 和 A. F. Molisch，《无线 D2D 网络中缓存的基本极限》，IEEE信息论汇刊，第 62 卷，第 849–869 页，2016年2月。[6] J. Dean 和 S. Ghemawat，《MapReduce：大型集群中的简化数据处理》，Communications of the ACM，第 51 卷，第 1 期，第 107–113 页，2008年。[7] N. Naderializadeh、M. A. Maddah-Ali 和 A. S. Avestimehr，《缓存辅助干扰管理的基本极限》，IEEE信息论汇刊，第 63 卷，第 3092–3107 页，2017年5月。[8] M. Chowdhury、M. Zaharia、J. Ma、M. I. Jordan 和 I. Stoica，《借助 Orchestra 管理计算机集群中的数据传输》，ACM SIGCOMM计算机通信评论，第 41 卷，第 4 期，第 98–109 页，2011年。[9] Z. Zhang、L. Cherkasova 和 B. T. Loo，《异构云环境中 MapReduce 作业的性能建模》，载于 2013年 IEEE 第六届云计算国际会议，第 839–846 页，IEEE，2013年。[10] S. Li、M. A. Maddah-Ali 和 A. S. Avestimehr，《编码 MapReduce》，载于 2015年第 53 届 Allerton 通信、控制与计算年会（Allerton），第 964–971 页，IEEE，2015年。[11] S. Li、M. A. Maddah-Ali、Q. Yu 和 A. S. Avestimehr，《分布式计算中计算与通信之间的基本权衡》，IEEE信息论汇刊，第 64 卷，第 1 期，第 109–128 页，2017年。[12] Y. H. Ezzeldin、M. Karmoose 和 C. Fragouli，《通信与分布式计算：另一种权衡曲线》，载于 2017年 IEEE 信息论研讨会（ITW），第 279–283 页，2017年11月。[13] Q. Yan、S. Yang 和 M. Wigger，《分布式计算中的存储、计算与通信权衡》，载于 2018年第 15 届无线通信系统国际研讨会（ISWCS），第 1–5 页，2018年8月。[14] Q. Yan、S. Yang 和 M. Wigger，《存储、计算与通信：分布式计算中的基本权衡》，载于 2018年 IEEE信息论研讨会（ITW），第 1–5 页，IEEE，2018年。[15] K. Lee、M. Lam、R. Pedarsani、D. Papailiopoulos 和 K. Ramchandran，《利用编码加速分布式机器学习》，IEEE信息论汇刊，第 64 卷，第 1514–1529 页，2018年3月。[16] S. Li、M. A. Maddah-Ali 和 A. S. Avestimehr，《面向存在掉队服务器的分布式计算的统一编码框架》，载于 2016年 IEEE Globecom 研讨会（GC Wkshps），第 1–6 页，2016年12月。[17] Q. Yan、M. Wigger、S. Yang 和 X. Tang，《存在掉队节点的分布式计算中的基本存储-通信权衡》，载于 2019年 IEEE信息论国际研讨会（ISIT），第 2803–2807 页，2019年7月。[18] V. Ramkumar 和 P. V. Kumar，《基于放置传递阵列的编码 MapReduce 方案》，载于 2019年 IEEE信息论国际研讨会（ISIT），第 3087–3091 页，2019年7月。[19] H. H. S. Chittoor、P. Krishnan、K. V. S. Sree 和 B. Mamillapalli，《基于射影几何的次指数和线性子分包编码缓存》，IEEE信息论汇刊，第 67 卷，第 9 期，第 6193–6222 页，2021年。[20] K. Shanmugam、M. Ji、A. M. Tulino、J. Llorca 和 A. G. Dimakis，《缓存辅助编码组播的有限长度分析》，IEEE信息论汇刊，第 62 卷，第 5524–5537 页，2016年10月。[21] Q. Yan、X. Tang、Q. Chen 和 M. Cheng，《通过二分图的强边染色设计放置传递阵列》，IEEE通信快报，第 22 卷，第 236–239 页，2018年2月。[22] Q. Yan、M. Cheng、X. Tang 和 Q. Chen，《集中式编码缓存方案的放置传递阵列设计》，IEEE信息论汇刊，第 63 卷，第 5821–5833 页，2017年9月。[23] M. Cheng、J. Jiang、Q. Yan 和 X. Tang，《具有灵活内存大小的编码缓存方案构造》，IEEE通信汇刊，第 67 卷，第 6 期，第 4166–4176 页，2019年。

[24] X. Zhong、M. Cheng 和 R. Wei，《具有线性子分包数的编码缓存方案》，IEEE通信汇刊，第69卷，第6期，页码3628–3637，2021年。[25] C. Shangguan、Y. Zhang 和 G. Ge，《集中式编码缓存方案：一种超图理论方法》，IEEE信息论汇刊，第64卷，页码5755–5766，2018年8月。[26] K. Shanmugam、A. M. Tulino 和 A. G. Dimakis，《利用 Ruzsa-Szemerédi 图实现线性子分包的编码缓存》，载于2017年IEEE国际信息论研讨会（ISIT），页码1237–1241，2017年。[27] L. Tang 和 A. Ramamoorthy，《基于线性分组码、减少子分包数的编码缓存方案》，IEEE信息论汇刊，第64卷，第4期，页码3099–3120，2018年。[28] J. Li 和 Y. Chang，《基于组合设计的放置传递阵列》，IEEE通信快报，第26卷，第2期，页码296–300，2022年。[29] X. Wu、M. Cheng、C. Li 和 L. Chen，《面向较小子分包数和灵活存储容量编码缓存的放置传递阵列设计》，IEEE通信汇刊，第70卷，第11期，页码7089–7104，2022年。[30] M. Cheng、J. Wang、X. Zhong 和 Q. Wang，《构造集中式编码缓存放置传递阵列的框架》，IEEE信息论汇刊，第67卷，第11期，页码7121–7131，2021年。[31] P. Krishnan，《基于二分图线图的编码缓存》，载于2018年IEEE信息论研讨会（ITW），页码1–5，2018年。[32] S. A. Saberali、L. Lampe 和 I. F. Blake，《无需文件分割的去中心化编码缓存》，IEEE无线通信汇刊，第18卷，页码1289–1303，2019年2月。[33] Q. Yan、X. Tang 和 Q. Chen，《放置传递阵列及其应用》，载于2018年IEEE信息论研讨会（ITW），页码1–5，IEEE，2018年。[34] Q. Yan、M. Wigger、S. Yang 和 X. Tang，《具有掉队节点的分布式计算中的基本存储-通信权衡》，IEEE通信汇刊，第68卷，第12期，页码7311–7327，2020年。[35] V. R. Aravind、P. K. Sarvepalli 和 A. Thangaraj，《用于线性子分包编码缓存的放置交付数组提升构造》，IEEE通信汇刊，第70卷，第12期，页码7817–7829，2022年。[36] J. Wang、M. Cheng、K. Wan 和 G. Caire，《一种通过笛卡尔积实现减少子分包数编码缓存的新框架》，载于2022年IEEE国际信息论研讨会（ISIT），页码1300–1305，2022年。[37] M. Cheng、Q. Yan、X. Tang 和 J. Jiang，《低速率和子分包数的编码缓存方案》，arXiv预印本 arXiv:1703.01548，2017年。[38] D. R. Stinson，《组合设计：构造与分析》。Springer Science & Business Media，2007年。[39] C. J. Colbourn 和 J. H. Dinitz，《组合设计手册》。CRC出版社，2006年。[40] P. Keevash，《设计的存在性》，arXiv预印本 arXiv:1401.3665，2014年。[41] J. Chibayi、D. Alila 和 F. Onyango，《非平凡t-设计的构造过程》，《美国理论与应用统计学杂志》，第6卷，第1期，页码52–60，2017年。[42] K. A. Osuolale 和 O. A. Otekunrin，《从仿射可分解的（v, b, r, k,λ）平衡不完全区组设计构造对称（（r+ 1）v, kr, kλ）平衡不完全区组设计的算法》，《年鉴：计算机科学系列》，第12卷，第2期，2014年。[43] H. Hanani，《关于横截设计》，载于《组合学》，页码43–53，Springer，1975年。[44] M. Braun、M. Kiermaier 和 A. Wassermann，《设计的q-类似物：子空间设计》，页码171–211。Cham：Springer国际出版公司，2018年。[45] M. Braun、T. Etzion、P. R. J. ¨Osterg˚ard、A. Vardy 和 A. Wassermann，《Steiner系统的 q-类比的存在性》，《数学论坛· Pi》，第4卷，文章e7，2016年。[46] T. Etzion 和 A. Vardy，《Steiner系统和覆盖设计的q-类比》，arXiv预印本 arXiv:0912.1503，2009年。[47] H. Suzuki，《 gf（2^m）上的2-设计》，《图与组合》，第6卷，页码293–296，1990年9月。[48] H. Suzuki，《GF(q)上的2-设计》，《图与组合》，第8卷，页码381–389，1992年12月。[49] A. Fazeli、S. Lovett 和 A. Vardy，《有限域上对所有t均存在非平凡t-设计》，《组合理论杂志》A辑，第127卷，页码149–160，2014年。[50] J. Hirschfeld，《有限域上的射影几何》。牛津数学专著丛书。牛津大学出版社，纽约，1998年。[51] A. Goel、M. Kapralov 和 S. Khanna，《正则二分图中在 $\mathcal{O}$（$n$log n）时间内求完美匹配》，《SIAM计算杂志》，第42卷，第3期，页码1392–1404，2013年。[52] D. Katyal、P. N. Muralidhar 和 B. S. Rajan，《基于交叉可分解设计的多接入编码缓存方案》，IEEE通信汇刊，第69卷，第5期，页码2997–3010，2021年。