with the appropriate IVs as inputs to compute the assigned output functions.

Throughout this paper, we consider the following design options. First, we assume each computing node computes all possible IVs from locally available files. This means that $M_k = |\mathcal{M}_k|$ represents both the number of files stored and number of map computations at node k. Second, we consider the design scenario such that each of the Q Reduce functions is computed exactly once ($s = 1$) at one node and $|\mathcal{W}_i \cap \mathcal{W}_j| = 0$ for $i \neq j$, where s is defined as the number of nodes which calculate each Reduce function.[2] Third, we consider the general scenario where each computing node can have a varying number of files, M_k and reduce functions, W_k.

This distributed computing network design yields two important performance parameters: the computation load, r, and the communication load, L. The computation load is defined as the number of times each IV is computed among all computing nodes, or $r = \frac{1}{N}\sum_{k=1}^{K} M_k$. In other words, the computation load is the number of IVs computed in the Map phase normalized by the total number of unique IVs, QN. The communication load is defined as the amount of traffic load (in bits) among all the nodes in the Shuffle phase normalized by QNT. Given the number of files mapped at each node, we are interested in the optimal communication load, L^*, over all possible file mappings, functions assignments and shuffle designs. [3]

Definition 1: Given the number of files at each node, $M_1, \ldots, M_K$, the optimal communication load is defined as

$$L^* \triangleq \inf\{L : (L, M_1, \ldots, M_K) \text{ is feasible}\}, \qquad (1)$$

where we consider all achievable file mapping, function assignments and shuffle designs.

III. Homogeneous Hypercube Computing Approach

In this section, we describe the proposed homogeneous CDC design based on the *hypercube* combinatorial structure. Our schemes are defined by *node grouping*, *file mapping*, *function assignment* and *shuffle method*. Two detailed examples, one for two-dimensional, and one for three-dimensional, are provided to illustrate the fundamental principles of the proposed design. These will be extended to the more general heterogeneous CDC scheme in Section IV.

In this section, we consider the scenario where the network is homogeneous. In other words, each node is assigned the same number of files and reduce functions. Also, every reduce function is computed exactly once at one node ($s = 1$).[4]

[2]The scenario of $s > 1$, meaning that each of the Q Reduce functions is computed at multiple nodes, is called *cascaded* distributed computing, introduced in [21]. In this paper, we do not consider this case.

[3]Note that we define L^* as a function of the number of files at each node, rather than just as a function of r as in previous CDC works. This excludes trivial designs that lead to $L = 0$. For instance, we can pick r nodes and assign the entire file library to each of these nodes. Furthermore, assign each function to one of the r nodes. In this case, since each node can compute all the necessary IVs itself, no Shuffle phase is required and $L = 0$. Our definition is more practical as it is tied to the number of computations (or mapped files) at each node.

[4]Our other work [31] focuses on the case of $s > 1$. The proposed schemes and analysis of [31] are significantly different from those presented in this work.

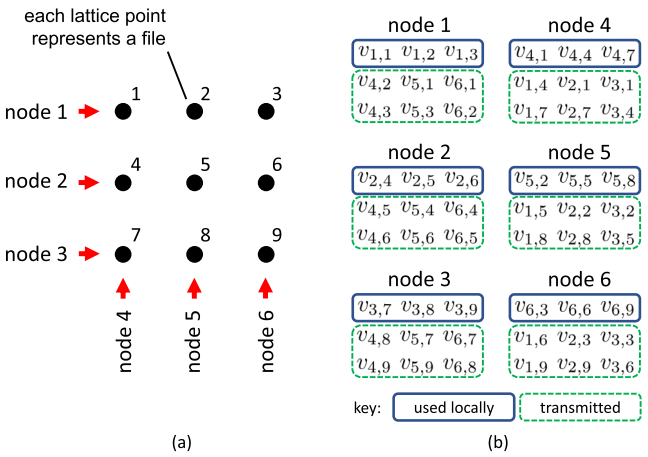

Fig. 1. (a) Lattice plane that defines file availability amongst the $K = 6$ computing nodes. Each lattice point represents a file and each node has a set of files available to it represents by a horizontal or vertical line of lattice points. (b) The IVs used locally and transmitted by each node.

Every node k computes a set of $W_k = \eta_2$ distinct functions and $Q = \eta_2\ K$ where $\eta_2 \in \mathbb{Z}^+$. The novel combinatorial hypercube design splits the nodes into r disjoint sets each of size $\frac{K}{r}$ and batches of η_1 files are assigned to one node from each set.[5] This is analogous to constructing a hypercube lattice of dimension r with the length of each side $\frac{K}{r}$ to describe the file placement at the nodes. We use this hypercube approach to better illustrate the examples of our new combinatorial design. We show that the required number of files is $N = \eta_1 \left(\frac{K}{r}\right)^r$ where $\eta_1 \in \mathbb{Z}^+$ and the number of multicasting groups is $G = \left(\frac{K}{r}\right)^r$. We first present a 2-dimension (a plane) example where $r = 2$.

A. 2-Dimension Example

In this example, we propose a distributed computing network based on a $r = 2$ dimensional hypercube (a plane) lattice where each side has length $\frac{K}{r} = 3$. There are $K = 6$ computing nodes each of which has access to $\frac{1}{3}$ of the file library. Each lattice point represents a file and each node has a set of files available to it represented by a line of lattice points as shown in Fig. 1(a). Specifically, there are two set of nodes: $\mathcal{K}_1 = \{1, 2, 3\}$ and $\mathcal{K}_2 = \{4, 5, 6\}$. Each node k in $\mathcal{K}_1$ (or $\mathcal{K}_2$) has access to $M_k = 3$ files, represented by three lattice points along a horizontal (or vertical) line. For instance, node 1 in $\mathcal{K}_1$ has access to three files w_1, w_2 and w_3 along the top horizontal line. Similarly, node 5 in $\mathcal{K}_2$ has access to three files w_2, w_5 and w_8, along the middle vertical line. Each node is responsible for computing one out of the $Q = 6$ reduce functions in the Reduce phase. More specifically, node k computes reduce function k and $W_k = 1$.

In the Map phase, nodes compute all $Q = 6$ IVs from each locally available file. Some IVs are necessary to compute the locally assigned reduce function. For example, as shown in Fig. 1(b), node 1 computes $v_{1,1}$, $v_{1,2}$ and $v_{1,3}$ and node 5 computes $v_{5,2}$, $v_{5,5}$ and $v_{5,8}$. These IVs do not have

[5]This scheme can be classified as a resolvable design for CDC, which was introduced in [27]. In addition, it also falls into the general framework of the Placement Delivery Array (PDA) designed for Device-to-Device coded caching [32].

to be transmitted and do not contribute to the communication load. However, other IVs are transmitted between nodes. We consider all possible pairs of nodes, termed node groups, consisting of one node from $\mathcal{K}_1$ and one node from $\mathcal{K}_2$. For instance, nodes 1 and 5 form node groups with each of the three nodes in $\mathcal{K}_2 = \{4, 5, 6\}$ or $\mathcal{K}_1 = \{1, 2, 3\}$, respectively. For the node group of $\{1, 5\}$, node 1 has computed $v_{5,1}$ and $v_{5,3}$ and transmits these IVs to node 5. Notice that, node 5 is incapable of computing these IVs itself because it does not have access to files w_1 and w_3. Similarly, node 5 has computed $v_{1,5}$ and $v_{1,8}$ and transmits these IVs to node 1 because node 1 does not have access to files w_1 and w_8. Fig. 1(b) also shows the IVs transmitted by each node. For example, node 1 will transmit IVs $v_{4,2}$ and $v_{4,3}$ to node 4, $v_{5,1}$ and $v_{5,3}$ to node 5 and $v_{6,1}$ and $v_{6,3}$ to node 6. On the other hand, node 1 will receive its requested IVs $v_{1,4}$ and $v_{1,7}$ from node 4, $v_{1,5}$ and $v_{1,8}$ from node 5, and $v_{1,6}$ and $v_{1,9}$ from node 6. Therefore, node 1 obtains all the IVs necessary for computing reduce function 1. In general, by considering all possible node groups, each node receives an IV for every file that it does not have. We can see this is true by recognizing that a node consecutively pairs with the three nodes in either $\mathcal{K}_1$ or $\mathcal{K}_2$, and the nodes in either $\mathcal{K}_1$ or $\mathcal{K}_2$ collectively have access to all the files.

Throughout this paper, we consider the case where each node computes all IVs from its available files similar to the original CDC work [21]. In this example, each IV is computed twice and $r = 2$ since each file is assigned to 2 nodes. In general, the computation load r is equivalent to the dimension of the hypercube which defines the file placement. Note that, nodes will compute some IVs that are never used to transmit, decode or compute a reduce function.[6] From 1(b) shows the IVs computed by each node that are utilized. Each node computes 3 IVs which are necessary for its own reduce function. Also, each node participates in 3 node pairs for which it needs to compute 2 IVs to transmit to the other node in the pair. In some applications, it may be possible for nodes to compute a select set of IVs to reduce the computation load as presented in [1], [23].

In this toy example, we only consider unicasting, therefore, the communication load is equivalent to the uncoded scenario and $L = \frac{2}{3}$, or the fraction of files not available at each node. This can be verified by recognizing that there are 9 pairs of nodes for which 2 IVs are transmitted from each node for each pair. In total, 36 of the 54 IVs are transmitted and $L = \frac{2}{3}$. In later examples, we will show how this scheme can be expanded to utilize coded multicasting and outperform the uncoded CDC scheme.

Remark 1: Interestingly, although the scheme generalized from this example is equivalent to unicast in this case, we observe that there actually exist multicasting opportunities in this example. For instance, node 1 could transmit $v_{4,2} \oplus v_{5,1}$ to nodes 4 and 5 (assuming that node 4 and 5 compute $v_{5,1}$ and $v_{4,2}$, respectively). In fact, all IVs could be transmitted in coded pairs where a node along one dimension transmits to 2 nodes aligned along the other dimensions, which would reduce the communication load by half.[7]

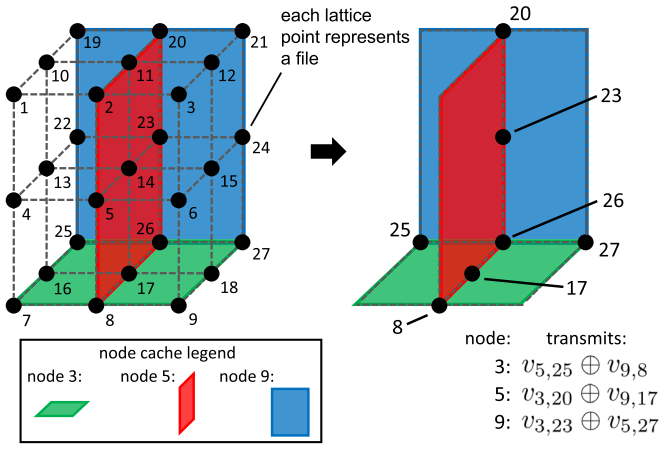

Fig. 2. (left) Cube lattice which defines file availability amongst the $K = 9$ computing nodes. Each lattice point represents a file and each node has set of files available to it represented by a plane of lattice points. The green, red and blue planes represent the files locally available to nodes 3, 5 and 9, respectively. (right) Intersections of planes which represent files that are locally available to multiple nodes and yields coded multicasting opportunities.

B. 3-Dimension Example

In the following, we provide a three-dimensional example and introduce notations used in the general scheme in Section III-C. We construct a computing network using a three-dimensional hypercube as shown in Fig. 2. Each lattice point in the cube, with its index $i \in$ [27] labeled next to the point, represents a different file set $\mathcal{B}_i = \{w_i\}$ which contains $\eta_1 = 1$ files. There are a total of $K = 9$ nodes, split into three node sets: $\mathcal{K}_1 = \{1, 2, 3\}$, $\mathcal{K}_2 = \{4, 5, 6\}$, and $\mathcal{K}_3 = \{7, 8, 9\}$, aligned along each of the $r = 3$ dimensions of the hypercube. Specifically, the three nodes in $\mathcal{K}_1 = \{1, 2, 3\}$ are represented by three parallel planes that go from top surface of the hypercube to the bottom. Node 3 is represented by the green plane that passes through lattice point 7. Node 1 and 2 are represented by the two planes (not shown) parallel to the green plane that go through lattice point 1 and point 4, respectively. The three nodes in $\mathcal{K}_2 = \{4, 5, 6\}$ are represented by three parallel planes that go from the left surface of the hypercube to the right. Node 5 is represented by the middle plane, shown in red, that goes through lattice point 8, and nodes 4 and 6 are represented by two planes (not shown) parallel to the red plane that go through lattice point 7 and 9, respectively. The nodes in $\mathcal{K}_3 = \{7, 8, 9\}$ are represented by three parallel planes that go from the front surface of the hypercube to the back. Node 9 is the blue plane, passing through lattice point 27, and nodes 7, 8 are represented by two planes (not shown) parallel to the blue plane and go through lattice points 9 and 18, respectively.

For file mapping, each node is assigned all the files indicated by the 9 lattice points on the corresponding plane. For instance, node 5, represented by the red plane, is assigned the file set

[6]We assume each node computes all Q IVs for each file. This is motivated by practical applications such as TeraSort or WordCount for which computing only a subset of IVs does not speed up overall MapReduce execution time.

[7]This is similar to the scheme outlined in [32], [33] for the analogous coded caching problem. However, as we will see for other examples and as discussed in [32], this scheme does not achieve a multiplicative gain for $r > 2$.

$\mathcal{M}_5 = \{w_2, w_5, w_8, w_{11}, w_{14}, w_{17}, w_{20}, w_{23}, w_{26}\}$. For each $i \in [3]$, the size of $\mathcal{K}_i$ is $\frac{K}{r} = 3$, which is the number of lattice points in the i-th dimension. Since the three nodes in each set $\mathcal{K}_i$ are aligned along dimension i, they collectively store the entire library of 27 files. Since each point i in the lattice is uniquely determined by the intersection of three planes, one from each dimension, the same point also represents a node group $\mathcal{T}_i$. For instance, node group $\mathcal{T}_{26} = \{3, 5, 9\}$ is represented by the three planes– green (node 3), red (node 5), and blue (node 9) intersecting at only one lattice point $i = 26$. It is clear that each file w_i is mapped to $r = 3$ nodes in $\mathcal{T}_i$. Each node k is assigned the $\eta_2 = 1$ functions of $\mathcal{W}_k = \{k\}$ because $Q = K = 9$ and each node k is only assigned the k-th reduce function.

In the Map phase, each node computes all IVs from locally available files. For example, node 5 will compute all possible IVs $\{v_{i,j} : i \in [Q], w_j \in \mathcal{M}_5\}$. The subset of IVs $\{v_{5,j} : w_j \in \mathcal{M}_5\}$ is used to calculate of the function output u_5. Furthermore, node 5 will use the subset of IVs in $\{v_{i,j} : i \in [Q] \setminus \mathcal{K}_2, w_j \in \mathcal{M}_5\}$ for transmission and decoding purposes when forming multicasting groups with nodes of $\mathcal{K}_1$ and $\mathcal{K}_3$. Note that, similar to the last example, node 5, and the other nodes, will compute some IVs that are not utilized.

We use the example of node group $\mathcal{T}_\alpha = \mathcal{T}_{26} = \{3, 5, 9\}$ to explain the Shuffle phase. Within node group $\mathcal{T}_{26}$, node 3 will multicast the summation of two IVs to nodes in $\mathcal{T}_{26} \setminus 3$, one intended for node 5, and one intended for node 9. The former must be available at both nodes 3 and 9, and the latter must be available at both nodes 3 and 5. To determine these IVs, we consider the set $\mathcal{V}^{\{5\}}_{\{3,9\}}$ and $\mathcal{V}^{\{9\}}_{\{3,5\}}$. The set $\mathcal{V}^{\{5\}}_{\{3,9\}}$ contains two IVs requested by node 5 that are available at nodes $3, 9$. To find these two IVs, we replace node 5 in $\mathcal{T}_{26}$ by one of the other two nodes in $\mathcal{K}_2$, which are nodes 4 and 6. This way, we obtain two node sets $\mathcal{T}_{25} = \{3, 4, 9\}$ and $\mathcal{T}_{27} = \{3, 6, 9\}$ that differ from $\mathcal{T}_\alpha$ only in the second element (the element that intersects $\mathcal{K}_2$). Thus, we define a line of lattice points, $\mathcal{L}_{5,\alpha} = \{25, 27\}$, representing the intersection of planes for nodes 3 and 9, but not points that intersect node 5's plane. This defines a set of IVs, $\mathcal{V}^{\{5\}}_{\{3,9\}} = \{v_{5,25}, v_{5,27}\}$, that are locally computed at nodes 3 and 9, but requested by node 5. Furthermore, we split $\mathcal{V}^{\{5\}}_{\{3,9\}}$ into two equally sized subsets, $\mathcal{V}^{\{5\},3}_{\{3,9\}} = \{v_{5,25}\}$ and $\mathcal{V}^{\{5\},9}_{\{3,9\}} = \{v_{5,27}\}$ which will be transmitted by nodes 3 and 9, respectively. Similarly, $\mathcal{L}_{9,\alpha} = \{8, 17\}$ is defined by the intersection of the planes for nodes 3 and 5, but not points intersecting node 9's plane. We now define $\mathcal{V}^{\{9\}}_{\{3,5\}} = \mathcal{V}^{\{9\},3}_{\{3,5\}} \bigcup \mathcal{V}^{\{9\},5}_{\{3,5\}} = \{v_{9,8}, v_{9,17}\}$ where $\mathcal{V}^{\{9\},3}_{\{3,5\}} = \{v_{9,8}\}$ will be transmitted by node 3 and $\mathcal{V}^{\{9\},5}_{\{3,5\}} = \{v_{9,17}\}$ will be transmitted by node 5. Once IV sets are found, node 3 transmits $\mathcal{V}^{\{5\},3}_{\{3,9\}} \oplus \mathcal{V}^{\{9\},3}_{\{3,5\}} = v_{5,25} \oplus v_{9,8}$ to nodes 5 and 9. Upon receiving this value, node 5 will subtract $v_{9,8}$ to recover $v_{5,25}$ and node 9 will subtract $v_{5,25}$ to recover $v_{9,8}$. The rest of the IV sets can be found in a similar fashion such that $\mathcal{V}^{\{3\}}_{\{5,9\}} = \mathcal{V}^{\{3\},5}_{\{5,9\}} \bigcup \mathcal{V}^{\{3\},9}_{\{5,9\}} = \{v_{3,20}, v_{3,23}\}$, and $\mathcal{V}^{\{9\}}_{\{3,5\}} = \mathcal{V}^{\{9\},3}_{\{3,5\}} \bigcup \mathcal{V}^{\{9\},5}_{\{3,5\}} = \{v_{9,8}, v_{9,17}\}$. Node 5 will transmit $v_{3,20} \oplus v_{9,17}$ to nodes 3 and 9. Node 9 will transmit $v_{3,23} \oplus v_{5,27}$ to nodes 3 and 5.

In this example, each node participates in 9 multicasting groups and transmits 1 coded message per group. Each transmission has the equivalent size of 1 IV. Therefore, the communication load is $L_{\mathrm{c}} = \frac{9 \cdot 9}{QN} = \frac{81}{9 \cdot 27} = \frac{1}{3}$, which is half of the uncoded communication load $L_{\mathrm{u}} = \frac{2}{3}$, or the fraction of files not available to each node.

C. General Homogeneous Scheme

In this subsection, we will introduce the general homogeneous scheme step by step as follows.

Node Grouping 1: Let $\mathcal{K} = \{1, 2, \cdots, K\}$ denote the set of K nodes. Assume that $\mathcal{K}$ is split into r equal-sized disjoint sets $\mathcal{K}_1, \ldots, \mathcal{K}_r$ that each contains $\frac{K}{r} \in \mathbb{Z}^+$ nodes. We define $\mathcal{T} \subset \mathcal{K}$ as a node group of size r if it contains exactly one node from each $\mathcal{K}_i$, i.e., $|\mathcal{T} \cap \mathcal{K}_i| = 1, \forall\, i \in [r]$. There are a total of $X = \binom{K}{r}^r$ possible node groups, denoted by $\mathcal{T}_1, \ldots, \mathcal{T}_X$. Furthermore, for each node group $\mathcal{T}_j$, we define its i-th component $\mathcal{T}_{j,i} = \mathcal{T}_j \cap \mathcal{K}_i$ as the node in $\mathcal{T}_j$ that is chosen from $\mathcal{K}_i$, where $i \in [r]$.

Node Group (NG) File Mapping: Given node groups $\mathcal{T}_1, \ldots, \mathcal{T}_X$, we split the N files into X disjoint sets labeled as $\mathcal{B}_1, \ldots, \mathcal{B}_X$. These file sets are of size $\eta_1 \in \mathbb{Z}^+$ and $N = \eta_1\, X$. Each file set $\mathcal{B}_i$ is only available to every node in $\mathcal{T}_i$. It follows that if node $k \in [K]$ belongs to a node group $\mathcal{T}_i$, then the file set $\mathcal{B}_i$ is available to this node. Hence, by considering all possible node groups $\mathcal{T}_i$ that node k belongs to, its available files, denoted by $\mathcal{M}_k$, are expressed as

$$\mathcal{M}_k := \bigcup_{i : k \in \mathcal{T}_i} \mathcal{B}_i. \tag{2}$$

Function Assignment 1: The Q reduce functions are split into K equal size, disjoint subsets labeled as $\mathcal{W}_1, \ldots, \mathcal{W}_K$. Each node is assigned $\eta_2 \in \mathbb{Z}^+$ reduce functions where $Q = \eta_2\, K$. For each $k \in [K]$, define $\mathcal{W}_k$ as the set of reduce functions assigned to node k.

Remark 2: By Node Grouping 1 and NG File Mapping, each node set $\mathcal{K}_i$ collectively maps the file library exactly once, and therefore, the file library is mapped r times among all K nodes. Note that, since each file belongs to a unique file set $\mathcal{B}_i$ and is mapped to a unique set of r nodes (in the node group $\mathcal{T}_i$), we must have $\frac{1}{N} \sum_{k=1}^{K} M_k = \frac{Nr}{N} = r$. Moreover, $M_k = \eta_1 \binom{K}{r}^{r-1}$ files are mapped to each node. Then, by Function Assignment 1, each node k is assigned $W_k = \eta_2$ reduce functions and each reduce function is assigned to exactly $s = 1$ node.

The Map, Shuffle and Reduce phases are defined as follows:

Map Phase: Each node $k \in [K]$ computes the set of IVs $\{v_{i,j} : i \in [Q], w_j \in \mathcal{M}_k\}$.

Node Group (NG) Shuffle Method: For every $\alpha \in [X]$, a coded message will be multicasted by each node $k \in \mathcal{T}_\alpha$ to serve independent requests of the rest $r - 1$ nodes in $\mathcal{T}_\alpha$. Here, each IV transmitted by node k is requested by a node $z \in \mathcal{T}_\alpha \setminus k$ and must be available to all other nodes in $\mathcal{T}_\alpha \setminus z$ to ensure that each node can decode successfully its own desired IVs from the broadcast. Next, we consider an arbitrary node group $\mathcal{T}_\alpha$ and a node $z \in \mathcal{T}_\alpha$. Assume that $z \in \mathcal{K}_h$, and thus $z = \mathcal{T}_{\alpha,h} = \mathcal{T}_\alpha \cap \mathcal{K}_h$. In the following, we fix the choice

of α, z, h and define

$$\mathcal{L}_{z,\alpha} = \{\ell \in [X] : \mathcal{T}_{\ell,h} \neq z, \mathcal{T}_{\ell,i} = \mathcal{T}_{\alpha,i}, \forall i \in [r] \setminus h\}. \quad (3)$$

Here, the set $\mathcal{L}_{z,\alpha}$ includes all indexes $\ell \in [X]$ such that the node group $\mathcal{T}_\ell$ differs from $\mathcal{T}_\alpha$ only in the h-th element, i.e., the node choice from $\mathcal{K}_h$. In other words, since $z \in \mathcal{K}_h$, then $\mathcal{T}_{\ell,h}$ can be any node in $\mathcal{K}_h$ except for z. Note that h is suppressed from the subscript of $\mathcal{L}_{z,\alpha}$ for notation simplicity. The definition of (3) ensures that for any $\ell \in \mathcal{L}_{z,\alpha}$, we have $z \notin \mathcal{T}_\ell$, but for any other node z' in $\mathcal{T}_\alpha \setminus z$, we have $z' \in \mathcal{T}_\ell$. This follows that while file set $\mathcal{B}_\ell$ is not mapped to node z, it is mapped to all other nodes z' in $\mathcal{T}_\alpha \setminus z$. Thus, we see that IVs of the type $\{v_{i,j}, i \in \mathcal{W}_z, w_j \in \mathcal{B}_\ell\}$ are requested by node z because z does not have $\mathcal{B}_\ell$, but are available to all nodes in $\mathcal{T}_\alpha \setminus z$ because they all have access to $\mathcal{B}_\ell$. This key idea is used to create multicast opportunities as follows. Formally, let us define

$$\mathcal{V}^{\{z\}}_{\mathcal{T}_\alpha \setminus z} = \bigcup_{\ell \in \mathcal{L}_{z,\alpha}} \{v_{i,j} : i \in \mathcal{W}_z, w_j \in \mathcal{B}_\ell\}, \quad (4)$$

which contains IVs requested by node z and are available at all nodes in $\mathcal{T}_\alpha \setminus z$. Furthermore, $\mathcal{V}^{\{z\}}_{\mathcal{T}_\alpha \setminus z}$ is split into $r-1$ disjoint subsets of equal size[8] denoted by $\mathcal{V}^{\{z\},\sigma_1}_{\mathcal{T}_\alpha \setminus z}, \ldots, \mathcal{V}^{\{z\},\sigma_{r-1}}_{\mathcal{T}_\alpha \setminus z}$ where $\{\sigma_1, \ldots, \sigma_{r-1}\} = \mathcal{T}_\alpha \setminus z$. Each node $k \in \mathcal{T}_\alpha$ sends the common multicast message

$$\bigoplus_{z \in \mathcal{T}_\alpha \setminus k} \mathcal{V}^{\{z\},k}_{\mathcal{T}_\alpha \setminus z} \quad (5)$$

to all nodes $z \in \mathcal{T}_\alpha \setminus k$.

Reduce Phase: For all $k \in [K]$, node k computes all output values u_q such that $q \in \mathcal{W}_k$.

Remark 3: For the homogeneous case, we have $|\mathcal{L}_{z,\alpha}| = \frac{K}{r} - 1$ because $\mathcal{T}_{\ell,h}$ can only be one of the $\frac{K}{r} - 1$ nodes in $\mathcal{K}_h \setminus z$. When using Node Grouping 1, NG File Mapping, and Function Assignment 1, in NG Shuffle Method we find each IV set, $\mathcal{V}^{\{z\}}_{\mathcal{T}_\alpha \setminus z}$, contains $\eta_1\eta_2\left(\frac{K}{r} - 1\right)$ IVs.

In the following, we will present a more complex 3-dimension example by accommodating the design procedures and all the notations introduced above.

D. Achievable Computation-Communication Load Trade-Off

The following theorem evaluates the trade-off between the computation and communication loads for the proposed scheme.

Theorem 1: By using Node Grouping 1, NG File Mapping, Function Assignment 1, and NG Shuffle Method, the communication load of the general homogeneous scheme is

$$L_{\mathrm{c}}(r) = \frac{K - r}{K\,(r-1)}. \quad (6)$$

Proof: Theorem 1 is proved in Appendix I as a special case of our general heterogeneous design which is defined in Section IV. ∎

[8]In general, $|\mathcal{V}^{\{z\}}_{\mathcal{T}_\alpha \setminus z}|$ may not be divisible by $r-1$, in which case the IVs of $\mathcal{V}^{\{z\}}_{\mathcal{T}_\alpha \setminus z}$ can be concatenated into a message and split into $r-1$ equal size segments. This process was presented in [21].

The optimality of this scheme is discussed in Section V by comparing the communication load of this scheme with that of the state-of-the-art scheme in [21].

Remark 4: When $s = 1$, coded caching [34], D2D coded caching [35], and CDC [21] are related. In particular, coded caching can be understood as a special case of CDC when each node is assigned one output function. This translates to a single file request in coded caching. Under this condition, in the original CDC work [21], the achievable design for $s = 1$ is equivalent to the D2D coded caching achievable design of [35]. The homogeneous CDC design presented in this paper is equivalent to the D2D coded caching design of [33]. However, CDC and coded caching differ in the following aspects. First, CDC allows the design of function assignment, while the file demand in coded caching is given. This yields different designs in both achievability and information theoretic converse. In the homogeneous case, the proposed hypercuboid CDC design shares some similarities but are different from the CDC schemes found in [36], [37] and the coded caching design in [38] when "translated" to the D2D setting using the approach introduced in [35].[9] This difference leads to a significant fact that the schemes in [36]–[38] cannot be extended to the heterogeneous settings. In contrast, the proposed hypercube homogeneous CDC design can be extended to the heterogeneous settings by allowing heterogeneous function assignments (see Section IV). The proposed scheme is applicable under some system parameters (e.g., $K/r \in \mathbb{Z}^+$) in the homogeneous setting. In practice, if it allows to assign only slightly different computation loads (e.g., number of output functions) to different nodes, the proposed coding gain can be preserved, meaning that the overall computation time can be significantly reduced compared to the uncoded approach. Second, in the general CDC framework, the file placement is uncoded in order to compute general functions. However, coding is allowed for the cache placement in coded caching.

IV. Heterogeneous Hypercuboid Computing Approach

In this section, we expand the proposed combinatorial hypercube design to accommodate heterogeneous computing networks. As mentioned in the introduction, one key novelty of our design is that nodes are assigned a varying number of files and reduce functions so that, nodes with larger local computation load, $\frac{M_k}{N}$, are assigned more reduce functions. In this case, the proposed heterogeneous design becomes a hypercuboid, consisting of P interleaved homogeneous hypercube networks. The homogeneous networks, $\mathcal{C}_p,\ \forall p \in [P]$, reflect hypercubes with different dimensions and lengths, representing distinct classes of nodes with varying local computation load. We start with an example and then present the general scheme.

A. 3-Dimension Hypercuboid Example

This example is presented in Fig. 3, where there are $P = 2$ classes of nodes $\mathcal{C}_1 = \mathcal{K}_1 \cup \mathcal{K}_2$ and $\mathcal{C}_2 = \mathcal{K}_3$ with different storage capability where $\mathcal{K}_1 = \{1, 2\}$, $\mathcal{K}_2 = \{3, 4\}$ and

[9]In fact, the coded caching scheme of [38] is equivalent to the CDC design of [27] under the "translation approach" in [35] when each node is assigned one output function, which is a single file.

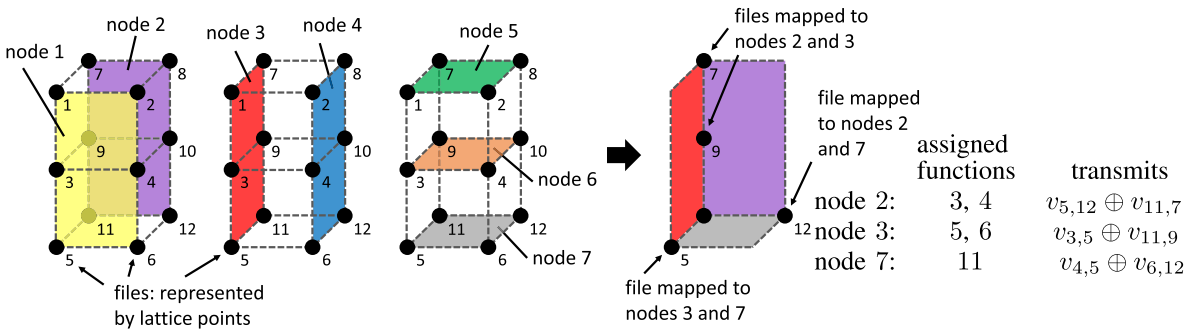

Fig. 3. Representations of a hypercuboid with $P = 2$, $r_1 = 2$, $m_1 = 2$, $r_2 = 1$ and $m_2 = 3$. *Left* 3 *cuboids*: Depictions of files mapped to nodes of $\mathcal{K}_1$, $\mathcal{K}_2$ and $\mathcal{K}_3$, respectively. *Right most cuboid*: Hypercuboid highlighting files stored at exactly 2 nodes of multicast group $\mathcal{T}_{11} = \{2, 3, 7\}$. These files, in addition to the assigned functions among these nodes, determine the IVs included in the coded multicasts, which are displayed to the right.

$\mathcal{K}_3 = \{5, 6, 7\}$. Each node in $\mathcal{C}_1$ stores half of the files and each node in $\mathcal{C}_2$ stores one-third of the files. Each node set, $\mathcal{K}_i$, collectively stores all $N = 12$ files. Each file is stored at the node group $\mathcal{T}_\alpha$ of 3 nodes such that it contains one node from each set $\mathcal{K}_1$, $\mathcal{K}_2$ and $\mathcal{K}_3$. For example, file w_1 is assigned to the nodes of $\mathcal{T}_1 = \{1, 3, 5\}$ and file w_{11} is assigned to the nodes of $\mathcal{T}_{11} = \{2, 3, 7\}$. All of the file placements are represented by the cuboid in Fig. 3. In the Map phase, the nodes will compute all IVs from their locally available files. Since every file is assigned to 3 nodes, the computation load is $r = 3$.

Different from previous works in CDC, nodes are assigned a varying number of reduce functions. We assign more reduce functions to nodes which map more files. Assume that there are $Q = 11$ reduce functions. We assign 2 reduce functions to each node of $\mathcal{K}_1$ and $\mathcal{K}_2$ and just 1 reduce function to each node of $\mathcal{K}_3$. Specifically, the function assignments are $\mathcal{W}_1 = \{1, 2\}$, $\mathcal{W}_2 = \{3, 4\}, \mathcal{W}_3 = \{5, 6\}, \mathcal{W}_4 = \{7, 8\}$, and $\mathcal{W}_5 = \{9\}, \mathcal{W}_6 = \{10\}$, and $\mathcal{W}_7 = \{11\}$. The reason we assigned this specific number of reduce functions to each node will become clear when we discuss the Shuffle phase.

In the Shuffle phase, the set of multicast groups includes all possible node groups $\mathcal{T}_\alpha$ which contain 1 node from each set $\mathcal{K}_1$, $\mathcal{K}_2$ and $\mathcal{K}_3$. Within each $\mathcal{T}_\alpha$, nodes send coded pairs of IVs to the other two nodes. For example, consider the node set $\mathcal{T}_\alpha = \mathcal{T}_{11} = \{2, 3, 7\}$. Following notations in Shuffle Method 1, we have $\mathcal{L}_{2,\alpha} = \{5\}$. This is because when replacing node $2 \in \mathcal{K}_1$ in $\mathcal{T}_\alpha$ by a different node in $\mathcal{K}_1$, we obtain $\mathcal{T}_5 = \{1, 3, 7\}$. Hence, using $\mathcal{W}_2 = \{3, 4\}$ and Eqn. (4), we obtain $\mathcal{V}_{3,7}^{\{2\}} = \{v_{3,5}, v_{4,5}\}$, which are IVs requested by node 2 and computed at nodes 3 and 7. Similarly, for node 3, we have $\mathcal{L}_{3,\alpha} = \{12\}$, and $\mathcal{V}_{2,7}^{\{3\}} = \{v_{5,12}, v_{6,12}\}$. For node 7, we have $\mathcal{L}_{7,\alpha} = \{7, 9\}$, corresponding to $\mathcal{T}_7 = \{2, 3, 5\}$ and $\mathcal{T}_9 = \{2, 3, 6\}$. While the size of $\mathcal{L}_{7,\alpha}$ is larger than that of $\mathcal{L}_{2,\alpha}$ and $\mathcal{L}_{3,\alpha}$, since $\mathcal{W}_7 = \{11\}$ is smaller, we obtain $\mathcal{V}_{2,3}^{\{7\}} = \{v_{11,7}, v_{11,9}\}$, which is the same size as that of $\mathcal{V}_{3,7}^{\{2\}}$ and $\mathcal{V}_{2,7}^{\{3\}}$. Using Eqn. (5), we see that nodes 2, 3, and 7 transmit $v_{5,12} \oplus v_{11,7}$, $v_{3,5} \oplus v_{11,9}$, and $v_{4,5} \oplus v_{6,12}$, respectively.

In this example, we see that by assigning a varying number of reduce functions to the nodes we can create symmetry among each node group, $\mathcal{T}_\alpha$, i.e., each node of the group requests the same number of IVs from the other nodes of the group. This symmetry can lead to savings in the communication load. Here, the communication load can be calculated by accounting for the $2 \cdot 2 \cdot 3 = 12$ node groups, where within each group, there are 3 transmissions of size T bits. By normalizing by QNT we find the communication load of the coded scheme is $L_{\text{c}} = \frac{36}{12 \cdot 11} = \frac{3}{11}$. We can compare this to the uncoded communication load, where each requested IV is transmitted alone. To compute the uncoded communication load, we count the number of IVs each node requests. Since the 4 nodes of $\mathcal{K}_1$ and $\mathcal{K}_2$ request $6 \cdot 2 = 12$ IVs each and the 3 nodes of $\mathcal{K}_3$ request 8 IVs each, we find $L_{\text{u}} = \frac{4 \cdot 12 + 3 \cdot 8}{12 \cdot 11} = \frac{6}{11}$. In this case, $L_{\text{c}} = \frac{1}{2} \cdot L_{\text{u}}$ since for the coded Shuffle policy every requested IV is transmitted in coded pairs. In the general heterogeneous CDC scheme proposed here, we will see that $L_c = \frac{1}{r-1} \cdot L_{\text{u}}$.

B. General Heterogeneous Scheme

In this subsection, we will introduce the general heterogeneous scheme step by step.

Node Grouping 2: The key idea of Node Grouping 2 is to form one heterogeneous network based on a hypercuboid design that consists of P interleaved homogeneous networks, represented by hypercubes of different dimensions r_p and sizes m_p within the hypercuboid. The K nodes consist of P disjoint sets denoted by $\mathcal{C}_1, \ldots, \mathcal{C}_P$, where $\sum_{p=1}^{P} |\mathcal{C}_p| = K$. For each $p \in [P]$, split $\mathcal{C}_p$ into $r_p \in \mathbb{Z}^+$ disjoint subsets, each of size m_p, denoted by $\{\mathcal{K}_{n_{p-1}+1}, \ldots, \mathcal{K}_{n_p}\}$, where $n_p = \sum_{i=1}^{p} r_i$. Hence, the entire network is comprised of r node sets, $\mathcal{K}_1, \ldots, \mathcal{K}_r$, where $r = \sum_{p=1}^{P} r_p$. Consider all possible node groups $\mathcal{T}_1, \ldots, \mathcal{T}_X$ of size r such that each contains one node from every node set $\mathcal{K}_1, \ldots, \mathcal{K}_r$, here $X = \prod_{i=1}^{r} |\mathcal{K}_i| = \prod_{p=1}^{P} m_p^{r_p}$. Denote $\mathcal{T}_{j,i} = \mathcal{T}_j \cap \mathcal{K}_i$, $\forall j \in [X]$ and $\forall i \in [r]$, as the node in $\mathcal{T}_j$ that is chosen from $\mathcal{K}_i$.

The file mapping is then determined by the NG File Mapping defined in Section III-C with node groups $\mathcal{T}_1, \ldots, \mathcal{T}_X$ defined by Node Grouping 2.

Remark 5: When using Node Grouping 2 and NG File Mapping, we form a hypercuboid made of P interleved hypercubes of different dimensions. For a given $p \in [P]$, $\mathcal{C}_p$ translates to r_p dimensions of size m_p of the hypercuboid. Moreover, $\mathcal{C}_p$ serves the role that is similar to that of a single hypercube of dimension r_p as in the homogeneous case. Specifically, $\mathcal{C}_p$ contains r_p node sets $\mathcal{K}_i$, each of size m_p. Here m_p is the number of lattice points along each dimension of the hypercube. The total number of nodes in $\mathcal{C}_p$ is thus $r_p m_p$. Nodes in each $\mathcal{K}_i$ collectively map the file library once. Hence, all nodes in $\mathcal{C}_p$ have the same storage

capacity that each maps a total of $\frac{N}{m_p}$ files. Collectively, nodes in $\mathcal{C}_p$ map the library r_p times. The P disjoint sets of $\mathcal{C}_1, \cdots, \mathcal{C}_p$ form one hypercuboid with r dimensions where there are r_p dimensions of size m_p for $p \in [P]$. Hence, each node group $\mathcal{T}_\alpha$ of size $r = \sum_{i=p}^{P} r_p$, defined in Node Group 2, consists of the union of P node groups, with size $r_1, \cdots, r_P$, respectively, chosen from each of the P interleved hypercubes corresponding to $\mathcal{C}_p, p \in [P]$. Note that, instead of each hypercube operating independently subject to its own computation load, r_p, the hypercuboid design takes full advantage of the total computation load, r, across the P hypercubes to achieve the gain of $\frac{1}{r-1}$ for the heterogeneous system.

Function Assignment 2: Define Y as the least common multiple (LCM) of $\{m_1 - 1, m_2 - 1, \ldots, m_P - 1\}$. Split the Q functions into K disjoint sets, labeled $\mathcal{W}_1, \ldots, \mathcal{W}_K$, where, in general, the sets may be different sizes. For each $k \in [K]$, $W_k = |\mathcal{W}_k| = \frac{\eta_2 Y}{m_p - 1}$ where $k \in \mathcal{C}_p$ and $\eta_2 \in \mathbb{Z}^+$ such that $Q = \eta_2 \, Y \sum_{p=1}^{P} \frac{r_p m_p}{m_p - 1}$. For each $k \in [K]$, let $\mathcal{W}_k$ be the set of reduce functions assigned to node k.

The Map and Shuffle phases follow our standard definition from Section III-C and the NG Shuffle Method is used for the Shuffle phase with node grouping defined by Node Grouping 2.

The correctness of the proposed heterogeneous CDC scheme is proved in Appendix III.

Remark 6: When using Node Grouping 2, NG File Mapping, Function Assignment 2, and NG Shuffle Method, we find that each intermediate value set $\mathcal{V}^{\{z\}}_{\mathcal{T}_\alpha \setminus z}$ contains $\eta_1 \eta_2 \, Y$ IVs.

Remark 7: Node Grouping 2 and Function Assignment 2 are a more general case of Node Grouping 1 and Function Assignment 1, respectively. Therefore, the homogeneous scheme of Section III-C is a special case of the general heterogeneous scheme here. By letting $P = 1$ such that $\mathcal{C}_1$ is the set of all nodes, we find $r = r_1$, $m_1 = \frac{K}{r}$, $X = \binom{K}{r}^r$ and $Y = \frac{K}{r} - 1$. Moreover, each node is assigned $\frac{\eta_2 \, Y}{m_1 - 1} = \eta_2$ reduce functions. For file availability, nodes are split into r disjoint, equal size sets, $\mathcal{K}_1, \ldots, \mathcal{K}_r$, and file sets of size η_1 are available to sets of nodes which contain exactly one node from each set $\mathcal{K}_1, \ldots, \mathcal{K}_r$.

Remark 8: Due to the constrained combinatorial structure, the proposed hypercuboid design will work only under certain heterogeneous individual memories and computation loads. The specific conditions for which our design is applicable are as follows. The number of files at each node must be of the form $M_k = \frac{N}{m}$ for some integer m. Then, the number of nodes which map $\frac{N}{m_1}$ files must be $c \cdot m_1$ for some integer c and m_1. Furthermore, the number of functions at each node is determined by the number of files at each node as defined by Function Assignment 2. In practice, we can group nodes with heterogeneous storage capacity and local computation load (i.e. $\frac{M_k}{N}$) to fit a hypercuboid design as close as possible (similar to "quantization") to take advantage of the system's heterogeneity.

An example is as follows. Consider $\frac{M_1}{N} = \frac{1}{2}, \frac{M_2}{N} = \frac{3}{5}, \frac{M_3}{N} = \frac{1}{3}, \frac{M_4}{N} = \frac{1}{3}, \frac{M_5}{N} = \frac{2}{5}$. The hypercuboid scheme is not applicable here because M_2 and M_5 do not take the form of $\frac{N}{m}$ for some integer m. To implement the hypercuboid scheme, we will round M_2 down to $\frac{N}{2}$ and M_5 down to $\frac{N}{3}$. Now there are 2 nodes that have $\frac{1}{2}$ of the files and 3 nodes with $\frac{1}{3}$ of the files. Then, note that the number of functions assigned to each node is a function of $M_1, \ldots, M_K$.

C. Achievable Trade-off Between Computation and Communication Loads

In this section, we first present a naive heterogeneous CDC design given the number of files and functions at each node. This naive approach simply uses an uncoded shuffle phase. We then expand it to find the communication load of an uncoded Shuffle phase, L_{u}, using the number of files and functions at each node from Node Grouping 2, NG File Mapping, Function Assignment 2 of the general heterogeneous scheme. Here, uncoded Shuffle phase means that all the requested IVs will be transmitted in a unicast fashion without coded multicasting. Note that, L_{u} represents the fraction of IVs which are requested by any node. Then, we demonstrate that the communication load using the the proposed hypercuboid scheme and the NG Shuffle Method is $L_{\mathrm{c}} = \frac{1}{r-1} \cdot L_{\mathrm{u}}$. Note that L_{u} and L_{c} are functions of $m_1, \ldots, m_P$ and $r_1, \ldots, r_P$, which specify the number of nodes and corresponding computation load in each node class of the heterogeneous computing network. Then, L_u and L_c are given in the following theorems.

Theorem 2: Given the number of files and functions at each node, $M_1, \ldots, M_K$ and $W_1, \ldots, W_K$, respectively, the following communication load is achievable using an uncoded shuffle phase

$$L_{\mathrm{u}} = \frac{1}{QN} \sum_{k \in [K]} W_k (N - M_k). \tag{7}$$

Corollary 1: By using Node Grouping 2, NG File Mapping, Function Assignment 2, and an uncoded Shuffle phase, the communication load is

$$L_{\mathrm{u}}(m_1, \ldots, m_P, r_1, \ldots, r_P) = \frac{r}{\sum_{p=1}^{P} \frac{r_p m_p}{m_p - 1}}. \tag{8}$$

Proof: Theorem 2 and Corollary 1 are proved in Appendix II. ∎

Next, we examine the communication load of the Shuffle phase with coded communication.

Theorem 3: By using Node Grouping 2, NG File Mapping, Function Assignment 2, and NG Shuffle Method, the communication load of the general heterogeneous scheme is

$$\begin{aligned} &L_{\mathrm{c}}(m_1, \ldots, m_P, r_1, \ldots, r_P) \\ &\quad = \frac{1}{r-1} \cdot \frac{r}{\sum_{p=1}^{P} \frac{r_p m_p}{m_p - 1}} \\ &\quad = \frac{1}{r-1} \cdot L_{\mathrm{u}}(m_1, \ldots, m_P, r_1, \ldots, r_P). \end{aligned} \tag{9}$$

Proof: Theorem 3 is proven in Appendix I. ∎

The communication load L_{c} is comprised of two parts: the local computing gain, L_{u}, and the global computing gain, $\frac{1}{r-1}$. The local computing gain represents the normalized number of IVs that must be shuffled. As nodes have access to a larger fraction of the files, the nodes will inherently request less in the Shuffle phase. The global computing gain stems from the

fact that with the coded design every transmission serves $r-1$ nodes with distinct requests.

D. Optimality

The information theoretic lower bound of the communication load derived in [21] is under the assumption of the homogeneous reduce function assignment.[10] Hence, it does not apply when reduce functions are heterogeneously assigned to the computing nodes. In the following, we use a cut-set bound based approach to derive a lower bound on the optimal communication load L^* as defined in (1). That is, given the number of files at each node, we find a lower bound on L^* over all possible uncoded file mappings, function assignments and shuffle designs. We use this bound to show that the proposed CDC design is optimal within a constant.

Theorem 4: Given the number of files mapped to each node, $M_1, \ldots, M_K$ we have

$$L^* \geq \min_{W_1,\ldots,W_K} \max_{\mathcal{S}\subseteq[K]} \frac{1}{QN}\left(N - \sum_{k\in\mathcal{S}} M_k\right) \sum_{k\in\mathcal{S}} W_k, \quad (10)$$

where $W_1, \ldots, W_K$ are the number of reduce functions assigned to each node and sum to Q.

Proof: Theorem 4 is proved in Appendix IV. ∎

Remark 9: Theorem 4 and its proof are motivated by the cut-set bound approach of the original coded caching approach [34], but with two key differences. First, the CDC problem is more general in that nodes require varying numbers of IVs because they are assigned varying numbers of reduce functions. Specifically, we consider a cut between the transmitted signal useful to a set of nodes and the functions collectively assigned to these nodes together with locally computed IVs. This is shown in (28) in Appendix IV. Second, we need to find the mutual information between the locally computed IVs and the IVs for assigned functions as shown in (29) in Appendix IV. Note that this term is not included in the coded caching cut-set bound because it is generally unknown.

Theorem 5: For a computing network with $M_1, \ldots, M_K$ and $W_1, \ldots, W_K$ defined by Node Grouping 2, NG File Mapping, and Function Assignment 2, and m_p is an even integer for all $p \in [P]$, we have

$$L_{\mathrm{c}} \leq \frac{4r}{r-1} L^*, \quad (11)$$

where L_{c}, given in (9), is the communication load achieved by using the hypercuboid design. The optimal communication load L^*, defined in (1), is over all possible file mappings, function assignments and shuffle designs given $M_1, \ldots, M_K$.

Proof: Theorem 5 is proved in Appendix V. ∎

V. Discussion

In this section, we will compare the performance of the proposed schemes to the state-of-the-art schemes in [21]. Specifically, we compare the required number of files, the required number of multicast groups and the communication load. When we compare the performance of the proposed heterogeneous CDC scheme with that of the homogeneous CDC in [21], we fix the computation load, r, the number of files, N, and the number of reduce functions, Q.[11]

[10]The converse of [21] for the homogeneous reduce function assignment was also derived in [39] using a different method.

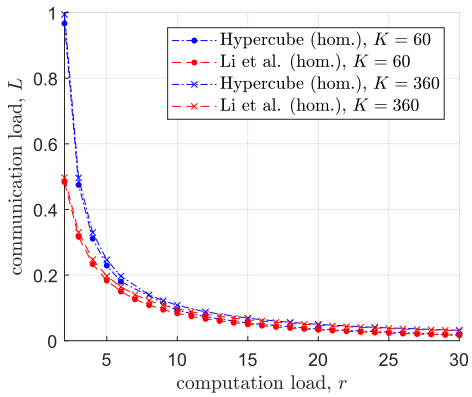

Fig. 4. A comparison of the resulting communication load for the newly proposed and the state-of-the-art homogeneous distributed computing schemes.

The scheme in [21] requires $N_1 = \binom{K}{r}\eta_1$ input files, $Q_1 = \binom{K}{s}\eta_2$ reduce functions. Moreover, the communication load as a function of K and r (with $s = 1$) is

$$L_1(r) = \frac{1}{r}\left(1 - \frac{r}{K}\right). \quad (12)$$

A. Homogeneous CDC

Using (6), we observe the following comparison

$$\frac{L_{\mathrm{c}}(r)}{L_1(r)} = \frac{rK}{K-r} \cdot \frac{K-r}{K\,(r-1)} = \frac{r}{r-1}. \quad (13)$$

For most values of r there is an insignificant increase in the communication load for the new combinatorial scheme and furthermore for $r \to \infty$ the two schemes yield the identical communication loads. Since our proposed homogeneous scheme uses the same function assignment as the scheme in [21], then this hypercube based design is asymptotically optimal in the information theoretic sense in general without fixing the file and function assignments. These findings are verified through simulation of the communication load as shown in Fig. 4.

While both schemes require the same number of outputs functions, $Q = K\eta_2$, the required number of input files has been drastically reduced in this case. It can be observed that the number of input files for the homogeneous hypercube design is $N_{\mathrm{c}} = \left(\frac{K}{r}\right)^r \eta_1$, while the scheme of [21] requires $N_1 = \binom{K}{r}\eta_1$ input files. Assuming $r = \Theta(K)$, by use of Stirling's formula to directly compare the two equations yields

$$\begin{aligned}
\frac{N_1}{N_{\mathrm{c}}} &= \frac{\binom{K}{r}}{\left(\frac{K}{r}\right)^r} = \frac{K!}{r!(K-r)!\left(\frac{K}{r}\right)^r} \\
&= \Theta\left(\frac{\sqrt{2\pi K}\left(\frac{K}{e}\right)^K}{2\pi\sqrt{r\,(\text{K-r})}\left(\frac{r}{e}\right)^r\left(\frac{K-r}{e}\right)^{(\text{K-r})}} \cdot \frac{1}{\left(\frac{K}{r}\right)^r}\right) \\
&= \Theta\left(\sqrt{\frac{K}{2\pi r(K-r)}} \cdot \left(\frac{K}{K-r}\right)^K\right) \\
&= \Theta\left(\sqrt{\frac{1}{K}} \cdot \left(\frac{K}{K-r}\right)^K\right). \quad (14)
\end{aligned}$$

[11]We adjust N and Q to be the same by using the appropriate η_1 and η_2.

When $r < K$, we find that (14) grows exponentially with K and, therefore, our proposed scheme has an exponential decrease in the number of required files.

As pointed out in [21], [27], the required number of multicast groups is also an important design parameter in CDC. If this number is large, it may take a long time to build such node groups such that the gain achieved by CDC is completely gone. It can be seen that the number of required multicast groups for the scheme in [21] is $U_1 = \binom{K}{r+1}$, while the required number of multicast group of the proposed scheme is $U_c = \left(\frac{K}{r}\right)^r$. Hence, by a similar computation to (14), it can be seen that

$$\frac{U_1}{U_c} = \Theta\left(\frac{r+1}{K-r} \cdot \sqrt{\frac{1}{K}} \cdot \left(\frac{K}{K-r}\right)^K\right), \tag{15}$$

which can grows exponentially with K such that the proposed hypercube scheme reduces the required number of multicast group exponentially.

Remark 10: The hypercube approach has similar performance compared to the CDC scheme based on the resolvable design proposed in [27], e.g., the required number of input files in [27] is $\left(\frac{K}{r}\right)^{r-1}$, which is slightly better than the proposed hypercube scheme. However, as discussed in Section IV, the proposed hypercube scheme can be extended to the heterogeneous CDC networks naturally while it is unclear how to extend the scheme in [27] to heterogeneous CDC networks.

B. Heterogeneous CDC

As shown in (9), the communication load of the proposed heterogeneous CDC design is $L_c(r) = \frac{1}{r-1} L_{\mathrm{u}}(r)$, where $\frac{1}{r-1}$ and $L_{\mathrm{u}}(r)$ are the global computing gain and the local computing gain, respectively. In comparison, for the homogeneous design in [21], we have $L_1(r) = \frac{1}{r}(1 - \frac{r}{K})$, where the global computing gain is $\frac{1}{r}$ and the local computing gain is $1 - \frac{r}{K}$. Next, we will show that even though the proposed heterogeneous design has an inferior global computing gain than that of [21] ($\frac{1}{r-1}$ versus $\frac{1}{r}$), it has a better local computing gain $L_{\mathrm{u}}(r) \leq (1 - \frac{r}{L})$, and hence can have a better communication load $L_c(r) < L_1(r)$ under certain parameter regimes.

Since $\sum_{p=1}^{P} \frac{r_p}{r} = 1$ and $\frac{m_p}{m_p - 1}$ is a convex function of m_p for $m_p > 1$, using (8) and Jensen's inequality, we can obtain

$$\begin{aligned}\frac{1}{L_{\mathrm{u}}(r)} &= \frac{\sum_{p=1}^{P} \frac{r_p m_p}{m_p - 1}}{r} = \sum_{p=1}^{P} \frac{r_p}{r} \cdot \frac{m_p}{m_p - 1} \\ &\geq \frac{\sum_{p=1}^{P} \frac{r_p m_p}{r}}{\left(\sum_{p=1}^{P} \frac{r_p m_p}{r}\right) - 1} = \frac{\frac{K}{r}}{\frac{K}{r} - 1} = \frac{K}{K - r},\end{aligned} \tag{16}$$

where $\sum_{p=1}^{P} r_p m_p = \sum_{p=1}^{P} |\mathcal{C}_p| = K$. Note that the inequality in (16) is strictly ">" if the network is truly heterogeneous, i.e., not all $\{m_p\}$ are equal. Hence,

$$L_{\mathrm{u}}(r) \leq \frac{K - r}{K} = 1 - \frac{r}{K}, \tag{17}$$

which shows that the local computing gain for our heterogeneous design is upper bounded by that of the homogeneous design in [21]. Using (9), we obtain,

$$L_{\mathrm{c}}(r) = \frac{1}{r-1} L_{\mathrm{u}}(r) \leq \frac{1}{r-1} \cdot \left(1 - \frac{r}{K}\right). \tag{18}$$

Thus, $L_{\mathrm{c}}(r)$ can be less than $L_1(r)$ for certain choices of r and K. For example, given a heterogeneous network defined by $m_1 = 2$, $r_1 = 4$ and $m_2 = 8$, $r_2 = 2$, we have $r = r_1 + r_2 = 6$, $K = r_1 m_1 + r_2 m_2 = 24$. We compare it with a homogeneous network with $r = 6$ and $K = 24$. The proposed heterogeneous design has a local computing gain of $L_{\mathrm{u}}(r) = \frac{7}{12} \approx 0.583$, which is less than that of the homogeneous design $1 - \frac{r}{K} = \frac{3}{4} = 0.75$, and a communication load of $L_{\mathrm{c}} = \frac{7}{60} \approx 0.117$, that is lower than that of the homogeneous design $L_1(r) = \frac{1}{8} = 0.125$.

Remark 11: We note that $L_{\mathrm{c}}(r) < L_1(r)$ is possible when there is a heterogeneous function assignment such that nodes are assigned varying numbers of functions. In [21], $L_1(r)$ was proved to be a lower bound on the communication load given r and K. However, the proof uses the implicit assumption that every node is assigned the same number of reduce functions. Our new finding is that if the reduce functions can be assigned in a heterogeneous fashion, then the communication load lower bound of [21] does not apply.

In Fig. 5, we provide additional comparisons of the communication load of the hypercuboid design and the homogeneous scheme of [21] with an equivalent computation load, r. Each design has a fixed number of nodes $K = 20$. The heterogeneous network is defined with $P = 2$ sets of nodes that map a different number of files and are assigned a different number of reduce functions. Specifically, $|\mathcal{C}_1| = 2(r-1)$ and $|\mathcal{C}_2| = K - 2(r-1)$ where $r_1 = r - 1$, $m_1 = 2$, $r_2 = 1$ and $m_2 = K - 2(r-1)$. In other words, the nodes of $\mathcal{C}_1$ each map $\frac{1}{2}$ of the files and the nodes of $\mathcal{C}_2$ each map a $\frac{1}{K-2(r-1)}$ fraction of the files which can be much less than $\frac{1}{2}$. Furthermore, nodes of $\mathcal{C}_1$ and $\mathcal{C}_2$ are each assigned $K - 2r + 1$ and 1 reduce functions, respectively. Fig. 5 shows that the communication load of the hypercuboid design is less than that of the state-of-the-art homogeneous design of [21] for $4 \leq r \leq 7$.

Comparisons for Large Networks: Next, we provide comparisons of the communication load of the proposed heterogeneous scheme and the homogeneous scheme [21] for networks with a large number of computing nodes K. We consider two cases.

Case 1: For the heterogeneous network, assume that $r_1, \ldots, r_P$ and r are fixed, but the fraction of files each node has access to, $\frac{1}{m_1}, \cdots, \frac{1}{m_P}$, decrease as K becomes large. Then, we have

$$\lim_{K \to \infty} L_{\mathrm{u}}(r) = 1 \quad \text{and} \quad \lim_{K \to \infty} \frac{L_{\mathrm{c}}(r)}{L_1(r)} = \frac{r}{r-1}. \tag{19}$$

In other words, $\frac{L_{\mathrm{c}}(r)}{L_1(r)} = \Theta(1)$.

Case 2: For the heterogeneous network, assume that $\frac{r_1}{K} = \beta_1, \ldots, \frac{r_P}{K} = \beta_K$ and $\frac{r}{K} = \beta$ are kept constant as K gets large. The fraction of files available to each node, $\frac{1}{m_1}, \cdots, \frac{1}{m_P}$, are also kept constant. It then follows from (16) that when the network is truly heterogeneous (not all $\{m_p\}$

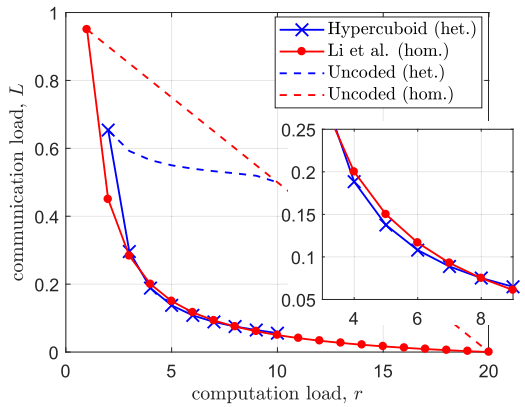

Fig. 5. A comparison of the proposed hypercuboid CDC design to the state-of-the-art CDC design of [21] with $K = 20$ nodes and an equivalent computation load, r. The heterogeneous hypercube is designed with parameters $r_1 = r-1, m_1 = 2, r_2 = 1$ and $m_2 = K-2(r-1)$. The hypercuboid design has a lower communication load than that of the homogeneous design for $4 \leq r \leq 7$. The communication load of the uncoded shuffle design is plotted for the homogeneous setting by the dashed red line. The communication load of the uncoded, naive design in Theorem 2 is plotted by the dashed blue line.

are equal), then we have

$$\lim_{K\to\infty} \frac{L_{\mathrm{c}}(r)}{L_1(r)} = \lim_{r\to\infty} \frac{r}{r-1} \cdot \frac{L_{\mathrm{u}}(r)}{1-\frac{r}{K}} = \frac{1}{1-\beta}\left(\frac{1}{\sum_{p=1}^{P} \frac{\beta_p}{\beta}\frac{m_p}{m_p-1}}\right) < \frac{1}{1-\beta}(1-\beta) = 1. \tag{20}$$

This means that for large networks considered here, the communication load of the proposed heterogeneous scheme is *strictly less* than that of the homogeneous scheme. Hence, for some heterogeneous file and computation load assignments, the fundamental trade-off proposed in [21] does not apply. This occurs because of the heterogeneous function assignment. As we discussed before, in the extreme case, where there exists a "super node" that can store all the files and compute all functions, the communication load is straightforwardly 0. However, for given heterogeneous storage capacities and computation loads, it is non-trivial to design an achievable CDC scheme such that its performance is superior compared to that of homogeneous CDC under the same total storage and computation load constraint.

For the hypercuboid design, the required number of files is $N = X = \prod_{p=1}^{P} m_p^{r_p}$ and reduce functions is $Q = Y\sum_{p=1}^{P} \frac{r_p m_p}{m_p-1}$ where Y is the LCM of $\{m_1-1,\ldots, m_P-1\}$. Unlike the homogeneous network case, due to the lack of CDC design for general heterogeneous networks, we cannot compare the proposed scheme to other schemes. Nevertheless, we believe that these numbers can serve as a benchmark for the future research in this topic.

VI. Conclusion and Future Directions

In this work, we introduced a novel hypercuboid combinatorial approach to design CDC for both homogeneous and heterogeneous distributed computing networks. This new design achieves a significant reduction in the number of files and functions compared to the state-of-the-art scheme in [21]. Moreover, the proposed schemes maintain a multiplicative computation-communication trade-off and are proven to be asymptotically optimal. Most importantly, we provided an explicit and systematic heterogeneous CDC design with optimality guarantees under certain network parameters. Surprisingly, we found that the optimal trade-off derived in [21] no longer applies when functions are heterogeneously assigned and as a result, the communication load of a heterogeneous network can be less than that of an equivalent homogeneous CDC network. We have also provided a converse for CDC when reduce functions are heterogeneously assigned. For the future research direction, first, it will be interesting to design other achievable schemes with heterogeneous function assignments. Second, it is challenging but important to find a tighter general information theoretic converse for communication load in heterogeneous CDC.

Appendix I
Proof of Theorems 1 and 3

For any $\alpha \in [X]$, and $z \in \mathcal{T}_\alpha$, where $z \in \mathcal{K}_h \subseteq \mathcal{C}_p$, it follows from Eq. (3), (4), and Remark 3 in Section III-C that

$$\left|\mathcal{V}^{\{z\}}_{\mathcal{T}_n\backslash z}\right| = |\mathcal{W}_z| \cdot \eta_1 \left|\mathcal{L}_{z,\alpha}\right| = |\mathcal{W}_z| \cdot \eta_1 (|\mathcal{K}_p| - 1) = \frac{\eta_2 Y}{m_p - 1} \cdot \eta_1 (m_p - 1) = \eta_1\eta_2 Y. \tag{21}$$

We consider X node groups of size r nodes, where for each group, every node of that group transmits a coded message of size $\left|\mathcal{V}^{\{z\}}_{\mathcal{T}_n\backslash z}\right|/(r-1)$, therefore, the communication load is

$$\begin{aligned} L_{\mathrm{c}}(m_1,\ldots,m_P,r_1,\ldots,r_P) &= \frac{1}{QN} \cdot X \cdot r \cdot \frac{\left|\mathcal{V}^{\{z\}}_{\mathcal{T}_n\backslash z}\right|}{r-1} \\ &= \frac{1}{\left(\eta_2 Y \sum_{p=1}^{P} \frac{r_p m_p}{m_p-1}\right)\eta_1 X} \cdot X \cdot r \cdot \frac{\eta_1\eta_2 Y}{r-1} \\ &= \frac{1}{r-1} \cdot \frac{r}{\sum_{p=1}^{P} \frac{r_p m_p}{m_p-1}}. \end{aligned} \tag{22}$$

For the homogeneous case where $P = 1$ and $\mathcal{C}_1 = [K]$, we find $r = r_1$, $m_1 = \frac{K}{r}$ and

$$L_{\mathrm{c}} = \frac{1}{r-1} \cdot \frac{r}{\sum_{p=1}^{P} \frac{r_p m_p}{m_p-1}} = \frac{1}{r-1} \cdot \frac{r}{\left(\frac{K}{\frac{K}{r}-1}\right)} = \frac{1}{r-1} \cdot \frac{K-r}{K}. \tag{23}$$

This completes the proof of Theorems 1 and 3.

Appendix II
Proof of Theorem 2 and Corollary 1

To prove Theorem 2, we count the number of IVs requested by each node. Consider node k which maps M_k files and is assigned W_k reduce functions. For each reduce function, the node requires an IV from each of the N files. However, this node only requests $N-M_k$ IVs for each assigned function since M_k of the IVs can be computed locally. Therefore,

node k requests $(N - M_k)W_k$ IVs, each of size T bits, to be transmitted unicast from the other nodes. Accounting for the IVs requested by all nodes and normalizing by QNT, we obtain Theorem 2.

To prove Corollary 1, we count the number of files and functions of each node and use Theorem 2 to find the uncoded communication load. For all $p \in [P]$, the number of files a node $k \in \mathcal{K}_j \subseteq \mathcal{C}_p$ has local access to is

$$M_k = \eta_1 \prod_{i \in [r] \backslash j} |\mathcal{K}_i| = \frac{\eta_1 X}{|\mathcal{K}_j|} = \frac{N}{m_p}. \tag{24}$$

Also, the number of functions assigned to a node $k \in \mathcal{K}_j \subseteq \mathcal{C}_p$ is $W_k = \frac{\eta_2 Y}{m_p - 1}$ where $Q = \eta_2 Y \sum_{p=1}^{P} \frac{r_p m_p}{m_p - 1}$. Using the result of (7), we find

$$\begin{aligned} &L_u(m_1, \ldots, m_P, r_1, \ldots, r_P) \\ &= \frac{1}{QN} \sum_{p \in [P]} \sum_{k \in \mathcal{C}_p} \frac{\eta_2 Y}{m_p - 1} \cdot \left(N - \frac{N}{m_p} \right) \qquad (25) \\ &= \frac{1}{Q} \sum_{p \in [P]} r_p m_p \frac{\eta_2 Y}{m_p - 1} \left(\frac{m_p - 1}{m_p} \right) \\ &= \frac{\eta_2 Y \sum_{p \in [P]} r_p}{\eta_2 Y \sum_{p=1}^{P} \frac{r_p m_p}{m_p - 1}} = \frac{r}{\sum_{p=1}^{P} \frac{r_p m_p}{m_p - 1}}, \qquad (26) \end{aligned}$$

where $|\mathcal{C}_p| = r_p m_p$ for all $p \in [P]$. Hence, we finished the proof of Corollary 1.

Appendix III
Correctness of Heterogeneous CDC Scheme

This proof includes 4 parts: 1) nodes only compute IVs from locally available files, 2) nodes only transmit locally computed IVs, 3) nodes can decode transmissions with requested IVs, 4) after the Map and Shuffle phases, nodes have all necessary IVs to compute their reduce functions.

For 1), any node $k \in [K]$ computes intermediate values of the set

$$\{v_{i,j} : i \in [Q], w_j \in \mathcal{M}_k\}. \tag{27}$$

In all cases $w_j \in \mathcal{M}_k$ for any $v_{i,j}$ computed by node k, therefore nodes only compute IVs from locally available files.

Next, we prove 2) and 3) simultaneously. Consider any node group $\mathcal{T}_\alpha$ and any node $k \in \mathcal{T}_\alpha$. We will prove that node k has access to multicast messages defined in Eq. (4) and (5). This is true because as discussed above Eq. (4), all nodes in $\mathcal{T}_\alpha \setminus z$, including node k, have access to the file set $\mathcal{B}_\ell$ where $\{\mathcal{T}_\alpha \setminus z\} \subset \mathcal{T}_\ell$. To see 3), when a node $z_0 \in \mathcal{T}_\alpha$ receives a multicast message from another node $k \in \mathcal{T}_\alpha$ in the form of Eq. (4), only one term, $\mathcal{V}_{\mathcal{T}_\alpha \setminus z}^{\{z_0\},k}$, is its desired message. The other terms are of the form $\mathcal{V}_{\mathcal{T}_\alpha \setminus z}^{\{z\},k}$, intended for node z, where $z \in \mathcal{T}_\alpha$ and, $z \neq z_0, k$. Since node $z_0 \in \mathcal{T}_\alpha \setminus z$, it has access to $\mathcal{V}_{\mathcal{T}_\alpha \setminus z}^{\{z\},k}$, and thus can decode its desired message correctly.

To prove 4), we need to show that for a given $z \in \mathcal{K}_h$, if some file $w_j \notin \mathcal{M}_z$, then node z must be able to recover its desired IVs $\{v_{i,j} : i \in \mathcal{W}_z\}$ from multicast messages of the form Eq. (4) and (5). To see this, assume that $w_j \in \mathcal{B}_{\ell_0}$. Consider node group $\mathcal{T}_{\ell_0}$. Since $w_j \in \mathcal{B}_{\ell_0}$ and $w_j \notin \mathcal{M}_z$, we must have $z \notin \mathcal{T}_{\ell_0}$. In other words, $\mathcal{T}_{\ell_0,h} \neq z$. Now, consider another node group $z \in \mathcal{T}_\alpha$ such that $\mathcal{T}_\alpha$ and $\mathcal{T}_{\ell_0}$ differ only in the h-th element: $\mathcal{T}_{\alpha,h} = z$ and $\mathcal{T}_{\alpha,i} = \mathcal{T}_{\ell_0,i}$ for any $i \neq h$. As defined in Eq. (4), since $\ell_0 \in \mathcal{L}_{z,\alpha}$ and $w_j \in \mathcal{B}_{\ell_0}$, node z will receive its desired IVs $\{v_{i,j} : i \in \mathcal{W}_z\}$ from multicast group messages of node group $\mathcal{T}_\alpha$ according to Eq. (4) and (5).

Appendix IV
Proof of Theorem 4

In this proof, we use the following notations: $X_\mathcal{K}$ represents the collection of all transmissions among all nodes, $\mathcal{W}_\mathcal{S}$ is the union of all functions assigned to nodes in $\mathcal{S}$, $\mathcal{M}_\mathcal{S}$ is the union of files available to nodes in $\mathcal{S}$, $V_{\mathcal{W}_\mathcal{S},:}$ is the set of IVs needed to compute the functions of $\mathcal{W}_\mathcal{S}$, and $V_{:,\mathcal{M}_\mathcal{S}}$ is the set of IVs computed from files of $\mathcal{M}_\mathcal{S}$. Here, we use ":" to denote all possible file or function indices.

Consider a particular given $W_1, \ldots, W_K$. Then, we consider some set of nodes $\mathcal{S} \subseteq [K]$ and use a cut-set to separate $X_\mathcal{K}$ and $V_{:,\mathcal{M}_\mathcal{S}}$ from $V_{\mathcal{W}_\mathcal{S},:}$ and obtain

$$\begin{aligned} I\left(V_{\mathcal{W}_\mathcal{S},:}; X_\mathcal{K}, V_{:,\mathcal{M}_\mathcal{S}}\right) &= H\left(V_{\mathcal{W}_\mathcal{S},:}\right) - H\left(V_{\mathcal{W}_\mathcal{S},:} | X_\mathcal{K}, V_{:,\mathcal{M}_\mathcal{S}}\right) \\ &\overset{(a)}{=} H\left(V_{\mathcal{W}_\mathcal{S},:}\right) \overset{(b)}{=} TN \sum_{k \in \mathcal{S}} W_k, \end{aligned} \tag{28}$$

where (a) follows from $H\left(V_{\mathcal{W}_\mathcal{S},:} | X_\mathcal{K}, V_{:,\mathcal{M}_\mathcal{S}}\right) = 0$ because given $X_\mathcal{K}$ and $V_{:,\mathcal{M}_\mathcal{S}}$, the IVs in $V_{\mathcal{W}_\mathcal{S},:}$ are known; (b) holds because there are $N \sum_{k \in \mathcal{S}} W_k$ IVs in $V_{\mathcal{W}_\mathcal{S},:}$, each of length T bits, and these IVs are assumed to be independent. Furthermore, note that, as defined in the problem definition, for two different nodes k_1 and k_2 we find $|\mathcal{W}_{k_1} \cap \mathcal{W}_{k_2}| = 0$ and each function is only assigned to 1 node. Next, notice that

$$\begin{aligned} &I\left(V_{\mathcal{W}_\mathcal{S},:}; V_{:,\mathcal{M}_\mathcal{S}}\right) \\ &= H\left(V_{\mathcal{W}_\mathcal{S},:}\right) - H\left(V_{\mathcal{W}_\mathcal{S},:} | V_{:,\mathcal{M}_\mathcal{S}}\right) \\ &\overset{(a)}{=} TN \sum_{k \in \mathcal{S}} W_k - T\left(N - |\cup_{k \in \mathcal{S}} \mathcal{M}_k|\right) \sum_{k \in \mathcal{S}} W_k \\ &= T |\cup_{k \in \mathcal{S}} \mathcal{M}_k| \sum_{k \in \mathcal{S}} W_k \leq T \sum_{k \in \mathcal{S}} M_k \sum_{k \in \mathcal{S}} W_k, \end{aligned} \tag{29}$$

where the second term in (a) corresponds to the entropy of the IVs that cannot be computed locally using files in $\mathcal{M}_\mathcal{S}$. Note that, each file produces 1 IV of size T bits for each function and the number of unique files among the nodes of $\mathcal{S}$ is at most $\sum_{k \in \mathcal{S}} M_k$. Thus, we have

$$\begin{aligned} &I\left(V_{\mathcal{W}_\mathcal{S},:}; X_\mathcal{K} | V_{:,\mathcal{M}_\mathcal{S}}\right) \\ &= H(X_\mathcal{K} | V_{:,\mathcal{M}_\mathcal{S}}) - H(X_\mathcal{K} | V_{\mathcal{W}_\mathcal{S},:}, V_{:,\mathcal{M}_\mathcal{S}}) \\ &\leq H\left(X_\mathcal{K}\right). \end{aligned} \tag{30}$$

By combining (30) with (28) and (29), we obtain

$$\begin{aligned} H\left(X_\mathcal{K}\right) &\geq I\left(V_{\mathcal{W}_\mathcal{S},:}; X_\mathcal{K} | V_{:,\mathcal{M}_\mathcal{S}}\right) \\ &= I\left(V_{\mathcal{W}_\mathcal{S},:}; X_\mathcal{K}, V_{:,\mathcal{M}_\mathcal{S}}\right) - I\left(V_{\mathcal{W}_\mathcal{S},:}; V_{:,\mathcal{M}_\mathcal{S}}\right) \\ &\geq TN \sum_{k \in \mathcal{S}} W_k - T \sum_{k \in \mathcal{S}} M_k \sum_{k \in \mathcal{S}} W_k \\ &= T \sum_{k \in \mathcal{S}} W_k \left(N - \sum_{k \in \mathcal{S}} M_k \right). \end{aligned} \tag{31}$$

By recognizing that $QNTL = H(X_{\mathcal{K}})$ and considering all possible node groups, $\mathcal{S}$, we obtain a lower bound on the communication load given $W_1, \ldots, W_K$. Then, as $W_1, \ldots, W_K$ are design parameters, we minimize over these parameters to obtain the bound of (10).

Appendix V
Proof of Theorem 5

The proof is based on the achievability in Theorem 3 and the converse bound in Theorem 4. First, define

$$L_{\mathcal{S}} \triangleq \frac{1}{QN}\left(N - \sum_{k\in\mathcal{S}} M_k\right)\sum_{k\in\mathcal{S}} W_k, \tag{32}$$

and

$$L_{\mathcal{S}^*} \triangleq \max_{\mathcal{S}\subset[K]} L_{\mathcal{S}}, \tag{33}$$

where $\mathcal{S}^* \subseteq [K]$ is the set of nodes that maximizes $L_{\mathcal{S}}$. Let $W_1^*, \ldots, W_K^*$ denote the number of function assignments at each node that minimize $L_{\mathcal{S}^*}$ given $M_1, \ldots, M_K$. We present two Claims below to characterize $W_1^*, \ldots, W_K^*$.

Claim 1: There exists an optimal solution $W_1^*, \ldots, W_K^*$ such that for any two nodes $k_1, k_2 \in [K]$, if $M_{k_1} = M_{k_2}$, then we must have $W_{k_1}^* = W_{k_2}^*$.

Proof: Assume that there exists a pair of nodes k_1 and k_2 such that $M_{k_1} = M_{k_2}$ and $W_{k_1}^* > W_{k_2}^*$. Let us consider a new function assignment with $W'_{k_1} = W'_{k_2} = \frac{W_{k_1}^* + W_{k_2}^*}{2}$. We will show that the new $L_{\mathcal{S}^*}$, corresponding to the new assignment, will not increase and thus, it gives an optimal assignment that satisfies Claim 1. To verify this, we will examine all $L_{\mathcal{S}}$ under the new assignment. First, note that for any group $\mathcal{S}$ that contains both k_1 and k_2, or contains none of them, $L_{\mathcal{S}}$ will not change if $W_{k_1}^*, W_{k_2}^*$ is changed to W'_{k_1}, W'_{k_2}. Next, we consider node groups that include only one of k_1 or k_2. Specifically, consider any node set $\mathcal{S}' \subseteq [K] \setminus \{k_1, k_2\}$ and define two sets $\mathcal{S}_1 = \{k_1\} \cup \mathcal{S}'$ and $\mathcal{S}_2 = \{k_2\} \cup \mathcal{S}'$. Note that $L_{\mathcal{S}^*} \geq \max\{L_{\mathcal{S}_1}, L_{\mathcal{S}_2}\}$ and $\max\{L_{\mathcal{S}_1}, L_{\mathcal{S}_2}\}$ will not increase if $W_{k_1}^*, W_{k_2}^*$ is changed to W'_{k_1}, W'_{k_2}. Therefore, $L_{\mathcal{S}^*}$ will not increase with this change. ∎

Next, to compare the bound of (10), we follow the proposed design to define $M_1, \ldots, M_K$ based on the parameters of hypercuboid $m_1, \ldots, m_P$ and $r_1, \ldots, r_P$. Note that, there are P sets of nodes $\mathcal{C}_p = \{k \in [K] : M_k = \frac{N}{m_p}\}$ where each node maps the same number of files. Moreover, we impose an additional requirement that m_p is an even integer as defined in Theorem 5.

Claim 2: Under the assumptions outlined in Theorem 5, we have $W_k^* = \frac{M_k}{rN}Q$ for all $k \in [K]$.

Proof: By claim 1, there exists an optimal assignment such that for each p, the nodes in $\mathcal{C}_p$ are assigned the same number of reduce functions. Define c_p as the ratio of the number of assigned functions to the number of files of each node in $\mathcal{C}_p$ (i.e., $W_k^* = c_p M_k$ where $k \in \mathcal{C}_p$). Consider a node group $\mathcal{S}_p \subset \mathcal{C}_p$ for some $p \in [P]$. We find, $L_{\mathcal{S}_p} = \frac{1}{QN}(N - M_{\mathcal{S}_p})c_p M_{\mathcal{S}_p}$ where $M_{\mathcal{S}_p} = \sum_{k\in\mathcal{S}_p} M_k$. Note that in this case $L_{\mathcal{S}_p}$ is maximized when $M_{\mathcal{S}_p} = \frac{N}{2}$, and this occurs when $\mathcal{S}_p$ contains $\frac{m_p}{2}$ nodes from $\mathcal{C}_p$. This is possible since m_p is an even integer and $\mathcal{C}_p$ contains at least m_p nodes. In summary, considering all subsets $\mathcal{S} \subset \mathcal{C}_p$, the maximum $L_{\mathcal{S}_p}$ is $L_{\mathcal{S}_p} = \frac{N}{4Q}c_p$.

Next, let $\hat{p} = \underset{p\in[P]}{\operatorname{argmax}}\ c_p$ be the index of the node set $\mathcal{C}_p$ with maximum c_p. Note that for any $\mathcal{S}$, we have $L_{\mathcal{S}} = \frac{1}{QN}(N - M_{\mathcal{S}})W_{\mathcal{S}}^*$ where $M_{\mathcal{S}} = \sum_{k\in\mathcal{S}} M_k$ and $W_{\mathcal{S}}^* = \sum_{k\in\mathcal{S}} W_k^*$. Using $W_k^* = c_p M_k \leq c_{\hat{p}} M_k$, we find that $W_{\mathcal{S}}^* \leq c_{\hat{p}} M_{\mathcal{S}}$. Thus, $L_{\mathcal{S}} \leq \frac{1}{QN}(N - M_{\mathcal{S}})c_{\hat{p}} M_{\mathcal{S}} \leq \frac{N}{4Q}c_{\hat{p}}$. It follows that $\mathcal{S}^*$, the node subset that maximizes $L_{\mathcal{S}}$, is $\frac{m_{\hat{p}}}{2}$ nodes from $\mathcal{C}_{\hat{p}}$. Hence, $L_{\mathcal{S}^*} = \frac{N}{4Q}c_{\hat{p}}$.

Finally, to minimize $L_{\mathcal{S}^*}$, we need to minimize $c_{\hat{p}} = \max_{p\in[P]} c_p$. We require that $Q = \sum_{p\in[P]} c_p \frac{N}{m_p}|\mathcal{C}_p| = N\sum_{p\in[P]} c_p r_p$, where node $k \in \mathcal{C}_p$ maps $\frac{N}{m_p}$ files and $|\mathcal{C}_p| = r_p m_p$. In this case, since $r = \sum_{p\in[P]} r_p$, we find $c_{\hat{p}} = c_1 = \ldots = c_P = \frac{Q}{rN}$. Therefore, for each node $k \in [K]$, we have $W_k^* = \frac{M_k}{rN}Q$. ∎

From Claim 2 and the converse bound of Theorem 4, we find that $L^* \geq \frac{1}{4r}$ given $M_1, \ldots, M_K$. This result applies to all possible file mappings, reduce function assignments and shuffle phase designs. This combines with $L_c < \frac{1}{r-1}$, shown in (18) obtained by Theorem 3, to give (11).

References

[1] N. Woolsey, R.-R. Chen, and M. Ji, "A new combinatorial design of coded distributed computing," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 726–730.

[2] N. Woolsey, R.-R. Chen, and M. Ji, "Coded distributed computing with heterogeneous function assignments," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–6.

[3] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Trans. Inf. Theory*, vol. 64, no. 3, pp. 1514–1529, Mar. 2018.

[4] S. Dutta, V. Cadambe, and P. Grover, "Short-dot: Computing large linear transforms distributedly using coded short dot products," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 2016, pp. 2100–2108.

[5] N. S. Ferdinand and S. C. Draper, "Anytime coding for distributed computation," in *Proc. 54th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2016, pp. 954–960.

[6] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," 2018, *arXiv:1801.07487*. [Online]. Available: http://arxiv.org/abs/1801.07487

[7] R. Bitar, P. Parag, and S. El Rouayheb, "Minimizing latency for secure distributed computing," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 2900–2904.

[8] C. Karakus, Y. Sun, S. Diggavi, and W. Yin, "Straggler mitigation in distributed optimization through data encoding," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5434–5442.

[9] W. Halbawi, N. Azizan-Ruhi, F. Salehi, and B. Hassibi, "Improving distributed gradient descent using Reed–Solomon codes," 2017, *arXiv:1706.05436*. [Online]. Available: http://arxiv.org/abs/1706.05436

[10] G. Suh, K. Lee, and C. Suh, "Matrix sparsification for coded matrix multiplication," in *Proc. 55th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Oct. 2017, pp. 1271–1278.

[11] A. Mallick, M. Chaudhari, U. Sheth, G. Palanikumar, and G. Joshi, "Rateless codes for near-perfect load balancing in distributed matrix-vector multiplication," 2018, *arXiv:1804.10331*. [Online]. Available: http://arxiv.org/abs/1804.10331

[12] R. K. Maity, A. S. Rawat, and A. Mazumdar, "Robust gradient descent via moment encoding with LDPC codes," in *Proc. SysML Conf.*, Jul. 2018, pp. 2734–2738.

[13] M. F. Aktas, P. Peng, and E. Soljanin, "Straggler mitigation by delayed Relaunch of tasks," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 45, no. 3, pp. 224–231, Mar. 2018.

[14] S. Wang, J. Liu, N. Shroff, and P. Yang, "Fundamental limits of coded linear transform," 2018, *arXiv:1804.09791*. [Online]. Available: http://arxiv.org/abs/1804.09791

[15] M. Ye and E. Abbe, "Communication-computation efficient gradient coding," 2018, *arXiv:1802.03475*. [Online]. Available: http://arxiv.org/abs/1802.03475
[16] K. Wan, H. Sun, M. Ji, and G. Caire, "Distributed linearly separable computation," 2020, *arXiv:2007.00345*. [Online]. Available: http://arxiv.org/abs/2007.00345
[17] M. A. Attia and R. Tandon, "Near optimal coded data shuffling for distributed learning," *IEEE Trans. Inf. Theory*, vol. 65, no. 11, pp. 7325–7349, Nov. 2019.
[18] A. Elmahdy and S. Mohajer, "On the fundamental limits of coded data shuffling for distributed machine learning," *IEEE Trans. Inf. Theory*, vol. 66, no. 5, pp. 3098–3131, May 2020.
[19] K. Wan, D. Tuninetti, M. Ji, G. Caire, and P. Piantanida, "Fundamental limits of decentralized data shuffling," *IEEE Trans. Inf. Theory*, vol. 66, no. 6, pp. 3616–3637, Jun. 2020.
[20] L. Chen, H. Wang, Z. Charles, and D. Papailiopoulos, "Draco: Byzantine-resilient distributed training via redundant gradients," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 902–911.
[21] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," *IEEE Trans. Inf. Theory*, vol. 64, no. 1, pp. 109–128, Jan. 2018.
[22] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.
[23] Y. H. Ezzeldin, M. Karmoose, and C. Fragouli, "Communication vs distributed computation: An alternative trade-off curve," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Nov. 2017, pp. 279–283.
[24] L. Song, S. R. Srinivasavaradhan, and C. Fragouli, "The benefit of being flexible in distributed computation," 2017, *arXiv:1705.08464*. [Online]. Available: http://arxiv.org/abs/1705.08464
[25] S. Prakash, A. Reisizadeh, R. Pedarsani, and A. S. Avestimehr, "Coded computing for distributed graph analytics," 2018, *arXiv:1801.05522*. [Online]. Available: http://arxiv.org/abs/1801.05522
[26] S. R. Srinivasavaradhan, L. Song, and C. Fragouli, "Distributed computing trade-offs with random connectivity," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 1281–1285.
[27] K. Konstantinidis and A. Ramamoorthy, "Resolvable designs for speeding up distributed computing," *IEEE/ACM Trans. Netw.*, vol. 28, no. 4, pp. 1657–1670, Aug. 2020.
[28] M. Kiamari, C. Wang, and A. Salman Avestimehr, "On heterogeneous coded distributed computing," in *Proc. GLOBECOM IEEE Global Commun. Conf.*, Dec. 2017, pp. 1–7.
[29] N. Shakya, F. Li, and J. Chen, "Distributed computing with heterogeneous communication constraints: The worst-case computation load and proof by contradiction," 2018, *arXiv:1802.00413*. [Online]. Available: http://arxiv.org/abs/1802.00413
[30] F. Xu and M. Tao, "Heterogeneous coded distributed computing: Joint design of file allocation and function assignment," 2019, *arXiv:1908.06715*. [Online]. Available: http://arxiv.org/abs/1908.06715
[31] N. Woolsey, R.-R. Chen, and M. Ji, "A combinatorial design for cascaded coded distributed computing on general networks," 2020, *arXiv:2008.00581*. [Online]. Available: http://arxiv.org/abs/2008.00581
[32] J. Wang, M. Cheng, Q. Yan, and X. Tang, "On the placement delivery array design for coded caching scheme in D2D networks," 2017, *arXiv:1712.06212*. [Online]. Available: http://arxiv.org/abs/1712.06212
[33] N. Woolsey, R.-R. Chen, and M. Ji, "Towards finite file packetizations in wireless device-to-device caching networks," *IEEE Trans. Commun.*, vol. 68, no. 9, pp. 5283–5298, Sep. 2020.
[34] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.
[35] M. Ji, G. Caire, and A. F. Molisch, "Fundamental limits of caching in wireless D2D networks," *IEEE Trans. Inf. Theory*, vol. 62, no. 2, pp. 849–869, Feb. 2016.
[36] Q. Yan, X. Tang, and Q. Chen, "Placement delivery array and its applications," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Nov. 2018, pp. 1–5.
[37] V. Ramkumar and P. V. Kumar, "Coded MapReduce schemes based on placement delivery array," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 3087–3091.
[38] Q. Yan, M. Cheng, X. Tang, and Q. Chen, "On the placement delivery array design for centralized coded caching scheme," *IEEE Trans. Inf. Theory*, vol. 63, no. 9, pp. 5821–5833, Sep. 2017.
[39] P. Krishnan, L. Natarajan, and V. Lalitha, "An umbrella converse for data exchange: Applied to caching, computing, shuffling & Rebalancing," 2020, *arXiv:2010.10459*. [Online]. Available: http://arxiv.org/abs/2010.10459

Nicholas Woolsey (Student Member, IEEE) received the B.S. degree in biomedical engineering from the University of Connecticut in 2012, the M.Eng. degree in bioengineering from the University of Maryland, College Park, in 2015, with a focus on signal processing, imaging, and optics, and the Ph.D. degree from the Department of Electrical and Computer Engineering, The University of Utah, in December 2020. From 2014 to 2017, he was an Electrical Engineer with Northrop Grumman Corporation (NGC), Ogden, UT, USA, developing test and evaluation methods, modernization solutions, and signal processing algorithms for the sustainment of aging aircraft and ground communication systems. He is currently with Trabus Technologies, San Diego, CA, USA, as a Signal Processing Engineer developing decentralized wireless network technologies. His research interests include combinatorial designs and algorithms for resource allocation, coding and efficient communications in distributed computing, and private and caching networks. He received the 2020 Best ECE Dissertation Award from The University of Utah for his Ph.D. degree, the Outside the Box Grant to investigate the design of a modern receiver that interfaces aging technology from NGC, and the 2016 Brent Scowcroft Team Award for performing exceptional systems engineering work.

Rong-Rong Chen (Member, IEEE) received the B.S. degree in applied mathematics from Tsinghua University, China, in 1993, and the M.S. degree in mathematics and the Ph.D. degree in electrical and computer engineering from the University of Illinois at Urbana–Champaign in 1995 and 2003, respectively. She was an Assistant Professor with The University of Utah from 2003 to 2011 and has been an Associate Professor since 2011. Her main research interests are in the area of communication systems and networks, with current emphasis on distributed computing, machine learning, caching networks, statistical signal processing, image reconstructions, and channel coding. She was a recipient of the M. E. Van Valkenburg Graduate Research Award for excellence in doctoral research with the ECE Department, University of Illinois at Urbana–Champaign, in 2003, and the prestigious National Science Foundation Faculty Early Career Development (CAREER) Award in 2006. She was rated among the Top 15% Instructors of the College of Engineering, The University of Utah, in 2017 and 2018. She has served as an Associate Editor for IEEE TRANSACTIONS ON SIGNAL PROCESSING and a Guest Editor for IEEE JOURNAL ON SELECTED TOPICS IN SIGNAL PROCESSING. She has served on the technical program committees of leading international conferences in wireless communication and networks.

Mingyue Ji (Member, IEEE) received the B.E. degree in communication engineering from the Beijing University of Posts and Telecommunications, China, in 2006, the M.Sc. degrees in electrical engineering from the KTH Royal Institute of Technology, Sweden, and the University of California, Santa Cruz, in 2008 and 2010, respectively, and the Ph.D. degree from the Ming Hsieh Department of Electrical Engineering, University of Southern California, in 2015. He subsequently was a Staff II System Design Scientist with Broadcom Corporation (Broadcom Ltd.) from 2015 to 2016. He is currently an Assistant Professor of the Electrical and Computer Engineering Department and an Adjunct Assistant Professor with the School of Computing, The University of Utah. He is interested in the broad area of information theory, coding theory, concentration of measure and statistics with the applications of caching networks, wireless communications, distributed storage and computing systems, distributed machine learning, and (statistical) signal processing. He received the IEEE Communications Society Leonard G. Abraham Prize for the Best IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS Paper in 2019, the Best Paper Award in IEEE ICC 2015 conference, the Best Student Paper Award in IEEE European Wireless 2010 Conference and the USC Annenberg Fellowship from 2010 to 2014. He has been serving as an Associate Editor of IEEE TRANSACTIONS ON COMMUNICATIONS since 2020.