Topological Coded Distributed Computing

Kai Wan*, Mingyue Ji†, Giuseppe Caire*
*Technische Universität Berlin, 10587 Berlin, Germany, {kai.wan, caire}@tu-berlin.de
†University of Utah, Salt Lake City, UT 84112, USA, mingyue.ji@utah.edu

GLOBECOM 2020 - 2020 IEEE Global Communications Conference | 978-1-7281-8298-8/20/$31.00 ©2020 IEEE | DOI: 10.1109/GLOBECOM42002.2020.9322245

***Abstract*—This paper considers the MapReduce-like coded distributed computing framework originally proposed by Li et al., which uses coding techniques when distributed computing servers exchange their computed intermediate values, in order to reduce the overall traffic load. In their original model, servers are connected via an error-free common communication bus allowing broadcast transmissions. However, this assumption is one of the major limitations for practical implementations since real-world data centers may have network topologies far more involved than a single broadcast bus. We formulate a topological coded distributed computing problem, where the computing servers communicate with each other through some switch network. By using a special instance of fat-tree topologies, referred to as *t-ary fat-tree* proposed by Al-Fares et al. which can be built by some inexpensive switches, we propose a coded distributed computing scheme to achieve the optimal max-link communication load (defined as the maximum load over all links) over any network topology.**

***Index Terms*—Coded distributed computing, network topology, fat-tree.**

I. Introduction

Recent years have witnessed the emergence of big data with wide range of applications. To cope with such a large dimension of data and the complexity of data mining algorithm, it is increasingly popular to use cloud computing infrastructures such as Amazon Web Services (AWS) [1], Google Cloud Platform [2], and Microsoft Azure [3]. In particular, modern distributed computing platforms such as MapReduce [4] and Spark [5] have attracted significant attentions since they enable the computation of large tasks on data sizes of order of terabytes. The path to exascale distributed computing poses a number of significant research challenges as distributed computing vendors attempt to deliver a hundred times performance improvement relative to today's distributed computing systems. While large scale distributed algorithms and simulations running at these extreme scales have the potential for achieving unprecedented levels of accuracy and providing dramatic insights into complex phenomena, they are also presenting new challenges. Keys among these are the challenges related to the computation and communication costs. In order to tackle these large-scale problems, it is critically important to understand the fundamental tradeoff between computation and communication. Inspired by the idea from the current development of coded caching networks [6], [7], the pioneering works [8], [9] introduces the concept of *Coded Distributed Computing* (CDC), which enables network coding among intermediate computed values to save significant communication load among servers. In particular, [9] studied the fundamental tradeoff between communication load and computation load in a "MapReduce-like" distributed computing system. Surprisingly, it showed that if each computation task is replicated at r servers, the total communication load $L(r)$ can be reduced by a factor r. This means that we can trade computation power for communication load, which has the potential to lead to a solution of the traffic congestion problem in the current distributed computing systems.

The framework considered in [9] contains *Map*, *Shuffle*, and *Reduce* phases. In the map phase, the K distributed computing servers process parts of the stored data locally and generate some intermediate values. In the shuffle phase, each server broadcasts some computed intermediate values to other servers through an error-free common-bus (each server can receive the packets transmitted by other servers without error), such that all the servers can obtain enough input values to compute the output functions in the reduce phase. Other aspects and extensions of CDC are considered in the literature such as reducing complexity [10]–[12], randomized connectivity [13], alternative metrics [14], and over wireless channels [15].

As pointed out in [9], [16], while the common-bus topology is meaningful for co-located processors, it is generally difficult to implement such topology for physically separated servers. Since the publications of [9], [16], designing a practical data center network topology that can reap the gains of coded distributed computing in terms of per-link communication load is widely open. In this paper, we consider a general switch network connecting the computation servers as illustrated in Fig. 1. Depending on the network topology, the max-link communication load may be more relevant compared to the total communication load sent from each server. In fact, eventually such max-link communication load may be the system bottleneck that will dominate the overall communication performance. Our objective is to find a practically used network topology and design appropriate coded distributed computing schemes, such that given the computation load in the map phase, the max-link communication load over all links in this topology (related to the communication latency) is minimized. In addition to the problem formulation, our main contributions in this paper are as follows.

- We characterize the optimal max-link communication load by proposing an information theoretic converse based on cut-set bounds and a coded distributed computing scheme on a single-switch topology.
- Motivated by the fact that a single switch topology is in general not scalable with the number of servers and very costly, because a powerful switch whose number of ports is equal to the number of servers K becomes more and

978-1-7281-8298-8/20/$31.00 ©2020 IEEE
Authorized licensed use limited to: The University of Utah. Downloaded on October 01,2021 at 16:54:17 UTC from IEEE Xplore. Restrictions apply.

more costly and impractical as K increases, we propose to use the t-ary fat-tree topology proposed in [17] as illustrated in Fig. 2 (detailed description on the topology will be provided in Section III-B), which has been widely used in practice for data center networks [18]. This t-ary fat-tree is built by some t-ports switches which can handle up to $\frac{t^3}{4}$ servers, and can significantly reduce the network building cost. By leveraging **the symmetry of the t-ary fat-tree network and the fact that there exist some paths between any two servers**, we then propose a coded distributed computing scheme based on the t-ary fat-tree, which achieves the optimal max-link communication load over any network topology.

To the best of the authors' knowledge this is the first paper showing that the whole distributed computing gain of [9] can be achieved in a different and much more practical topology rather than the oversimplified and impractical common-bus topology. Therefore, this is an important step in the direction of bringing coded distributed computing much closer to practice.

Notation convention: Calligraphic symbols denote sets and sans-serif symbols denote system parameters. We use $|\cdot|$ to represent the cardinality of a set or the length of a vector; $[a:b] := \{a, a+1, \ldots, b\}$ and $[n] := [1, 2, \ldots, n]$.

II. System Model

A. Coded Distributed Computing Problem in [9]

We first briefly review the coded distributed computing problem in [9], which aims to compute Q arbitrary output values (denoted by $u_1, \ldots, u_{\mathsf{Q}}$) from N input data files (denoted by $w_1, \ldots, w_{\mathsf{N}}$) using a cluster of K distributed servers. For some $\mathsf{s} \in [\mathsf{K}]$ where $\binom{\mathsf{K}}{\mathsf{s}}$ divides Q, it is required that each subset of s servers compute a disjoint subset of $\frac{\mathsf{Q}}{\binom{\mathsf{K}}{\mathsf{s}}}$ output values. The set of output values which server k needs to compute is denoted by $\mathcal{W}_k$. The computation proceeds in three phases: *Map*, *Shuffle*, and *Reduce*.

Map phase. Each server $k \in [\mathsf{K}]$ computes the Map functions of data files in $\mathcal{M}_k \subseteq [\mathsf{N}]$, where $\mathcal{M}_k$ is stored in its memory. For each data file w_n where $n \in \mathcal{M}_k$, server k computes $g(w_n) = (v_{1,n}, \ldots, v_{\mathsf{Q},n})$, where $v_{q,n}$ is an intermediate value represented by T information bits for each $q \in [\mathsf{Q}]$. The output value u_q where $q \in [\mathsf{Q}]$ can be directly computed from some intermediate values, i.e., $u_q := h_q(v_{q,1}, \ldots, v_{q,\mathsf{N}})$ for some function h_q. The computation load, defined as

$$r = \frac{\sum_{k\in[\mathsf{K}]} |\mathcal{M}_k|}{\mathsf{N}} \in [1, \mathsf{K}],$$

represents the average number of nodes that map each data file.

Shuffle phase. To compute the output value u_q where $q \in \mathcal{W}_k$, in the shuffle phase server k needs to recover the intermediate values $\{v_{q,n} : n \notin \mathcal{M}_k\}$, which are not computed by itself in the map phase. For this purpose, each server k creates an ℓ_k-bit message X_k based on its computed intermediate values in the map phase, i.e.,

$$X_k = \psi(\{g(w_n) : n \in \mathcal{M}_k\}).$$

The message X_k is then broadcasted from server k to other servers through a common communication bus. The communication load, denoted by

$$L = \frac{\sum_{k\in[\mathsf{K}]} \ell_k}{\mathsf{QNT}},$$

represents the normalized number of bits communicated in the system.

Reduce phase. Each server $k \in [\mathsf{K}]$ first decodes the intermediate values $\{v_{q,n} : n \notin \mathcal{M}_k\}$ from the received messages $\{X_j : j \in [\mathsf{K}] \setminus \{k\}\}$, and then computes the output value $u_q := h_q(v_{q,1}, \ldots, v_{q,\mathsf{N}})$ for each $q \in \mathcal{W}_k$.

The objective is to design the Map, Shuffle and Reduce phases such that the communication load $L(r)$ is minimized given the computation load $r \in [1, \mathsf{K}]$.

It was proved in [9, Theorem 2] that the optimal tradeoff $L^\star(r)$ is the lower convex envelop of the following points,

$$L^\star(r) = \sum_{t=\max\{r+1,\mathsf{s}\}}^{\min\{r+\mathsf{s},\mathsf{K}\}} \frac{t\binom{\mathsf{K}}{t}\binom{t-2}{r-1}\binom{r}{t-\mathsf{s}}}{r\binom{\mathsf{K}}{r}\binom{\mathsf{K}}{\mathsf{s}}}, \tag{1}$$

where $r \in [\mathsf{K}]$.

The achievable scheme in [9, Section V] is based on linear coding. Define T_k as the transmitted message by server k in the achievable scheme [9] for each $k \in [\mathsf{K}]$. T_k contains

$$\frac{L^\star(r)\mathsf{QNT}}{\mathsf{K}}, \tag{2}$$

bits, and could be written as a set of non-overlapping sub-messages,

$$T_k = \left\{T_k^{\mathcal{S}} : \mathcal{S} \subseteq [\mathsf{K}] \setminus \{k\}\right\},$$

where $T_k^{\mathcal{S}}$ represents the sub-messages transmitted by server k which are useful to servers in $\mathcal{S}$. For each server $j \in [\mathsf{K}]$, the set of received sub-messages which are useful to server j is defined as

$$U_j = \left\{T_k^{\mathcal{S}} : k \in [\mathsf{K}] \setminus \{j\}, \mathcal{S} \subseteq [\mathsf{K}] \setminus \{k\}, j \in \mathcal{S}\right\}.$$

An important observation from the achievable scheme in [9] is that

$$|U_j| = \left(\mathsf{N} - \frac{r\mathsf{N}}{\mathsf{K}}\right)\frac{\mathsf{QsT}}{\mathsf{K}}, \tag{3}$$

which is equal to the number of bits in $\{v_{q,n} : q \in \mathcal{W}_j, n \notin \mathcal{M}_j\}$ (the total length of the intermediate values to compute u_q for all $q \in \mathcal{W}_j$, which are not computed by server j in the map phase). For the sake of space limitation we invite the reader to refer to [9] for more details.

B. Topological Coded Distributed Computing

The coded distributed computing framework in [9] assumes that each server broadcasts some messages to others through a common communication bus. However, this topology is not used in practice. Instead, we always need to build a topological network composed of switches and wired links to enable the communications among the servers. Fig. 1 illustrates the general data center networks considered in this paper, where

Authorized licensed use limited to: The University of Utah. Downloaded on October 01,2021 at 16:54:17 UTC from IEEE Xplore. Restrictions apply.

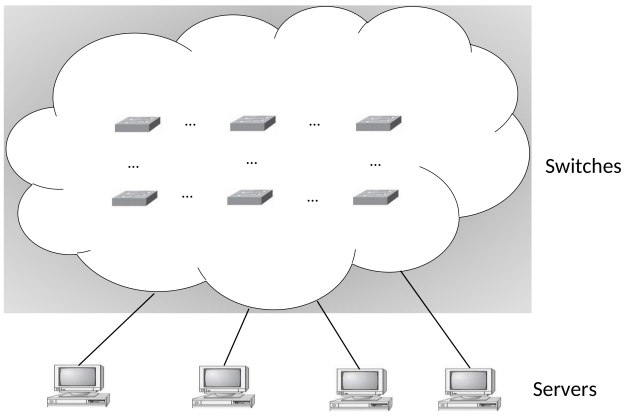

Fig. 1: The topological coded distributed computing problem.

each server is connected to a cloud of switches through an individual wired link. The switches in the network cannot compute functions, and we assume that there does not exist any direct link from one server to another.

The map and reduce phases in our considered problem are the same as in [9]. In the shuffle phase, instead of assuming the common communication bus, we need to design the topology in the system. Assume there are totally V wired links in the designed topology.[1] For each link $v \in [V]$, the number of uplink (i.e., from the bottom to the top in the topology) transmitted bits through link v is denoted by R_v^{up}, and the number of downlink (i.e., from the top to the bottom) transmitted bits through link v is denoted by R_v^{down}. The total number of bits transmitted through link v is denoted by

$$R_v = R_v^{\text{up}} + R_v^{\text{down}}.$$

We define the max-link communication load D as the maximal normalized number of bits transmitted through each link, where

$$D = \max_{v\in[V]} \frac{R_v}{\mathsf{QNT}}.$$

The objective is to design a network topology and an achievable scheme to characterize the communication load $D^\star(r)$ given the computation load $r \in [1, \mathsf{K}]$.

III. Main Results

A. Optimal max-link communication load

Theorem 1. *For the considered topological coded distributed computing problem, the optimal tradeoff between the max-link communication load and the computation load is the lower convex envelop of the following points,*

$$D^\star(r) = \frac{L^\star(r)}{\mathsf{K}} + \frac{\mathsf{s}}{\mathsf{K}}\left(1 - \frac{r}{\mathsf{K}}\right),\ \forall r \in [\mathsf{K}]. \tag{4}$$

[1]Notice that we consider here both the links from servers to the interconnection switching network and possible internal links between the switches.

Proof: Achievability. First, we need to design a network topology. We simply let all servers be connected to one switch at the top. Recall that T_k where $k \in [\mathsf{K}]$ is the transmitted message by server k in the achievable scheme [9]. Each server k transmits T_k to the top switch. The switch forwards U_j to each server $j \in [\mathsf{K}]$. The lengths of T_k and U_j are given in (2) and (3), respectively. Hence, we prove that the communication load in (4) is achieved.

Converse. Denote the index of the link directly connected to server k by v_k. We first consider the uplink transmission from the servers. The total uplink load through the links in $\{v_1, \ldots, v_{\mathsf{K}}\}$ should be no less than $L^\star(r)$, i.e.,

$$\sum_{k\in[\mathsf{K}]} R_{v_k}^{\text{up}} \geq L^\star(r). \tag{5}$$

We then consider the downlink transmission from the cloud to the servers. Recall that in the shuffle phase server k needs to recover $\{v_{q,n} : q \in \mathcal{W}_k, n \notin \mathcal{M}_k\}$, and that $|\mathcal{W}_k| = \frac{\mathsf{Q}}{\binom{\mathsf{K}}{\mathsf{s}}}\binom{\mathsf{K}-1}{\mathsf{s}-1} = \frac{\mathsf{Qs}}{\mathsf{K}}$. In other words, server k needs to recover $\frac{\mathsf{Qs}}{\mathsf{K}}(\mathsf{N} - |\mathcal{M}_k|)$. Hence, the total number of intermediate values needed to be recovered by all servers is

$$\sum_{k\in[\mathsf{K}]} \frac{\mathsf{Qs}}{\mathsf{K}}(\mathsf{N} - |\mathcal{M}_k|) = \frac{\mathsf{Qs}}{\mathsf{K}}(\mathsf{NK} - \mathsf{N}r). \tag{6}$$

Recall that the length of each intermediate value is T. From (6), the total downlink load through the links in $\{v_1, \ldots, v_{\mathsf{K}}\}$ can be bounded as follows,

$$\begin{aligned}\sum_{k\in[\mathsf{K}]} R_{v_k}^{\text{down}} &\geq \frac{\mathsf{Qs}}{\mathsf{K}}(\mathsf{NK} - \mathsf{N}r)\frac{\mathsf{T}}{\mathsf{QNT}} \\ &= \mathsf{s}\left(1 - \frac{r}{\mathsf{K}}\right).\end{aligned} \tag{7}$$

From (5) and (7), we have

$$\begin{aligned}\max_{k\in[\mathsf{K}]} R_{v_k} &\geq \frac{1}{\mathsf{K}}\sum_{k\in[\mathsf{K}]}(R_{v_k}^{\text{up}} + R_{v_k}^{\text{down}}) \\ &\geq \frac{L^\star(r)}{\mathsf{K}} + \frac{\mathsf{s}}{\mathsf{K}}\left(1 - \frac{r}{\mathsf{K}}\right),\end{aligned}$$

which coincides (4). Hence, we prove Theorem 1. ■

The optimal max-link communication load could be achieved by using a single switch connected to all the servers. However, we need a giant switch of K ports, which is much more expensive than a network with small switches (see [17]). One important question to ask is whether there exists a topology formed only by switches with a number of ports significantly smaller than K, and yet achieving the same optimal load in (4).

B. Description of t-ary Fat-tree in [17]

We can answer the question above by using the t-ary fat-tree topology proposed in [17] (illustrated in Fig. 2). As a matter of fact, the fat tree architecture is viable and practical interconnection architecture in data centers and enjoys the property of scalability since it uses only switches with a constant number of ports independent of the number of nodes

Authorized licensed use limited to: The University of Utah. Downloaded on October 01,2021 at 16:54:17 UTC from IEEE Xplore. Restrictions apply.

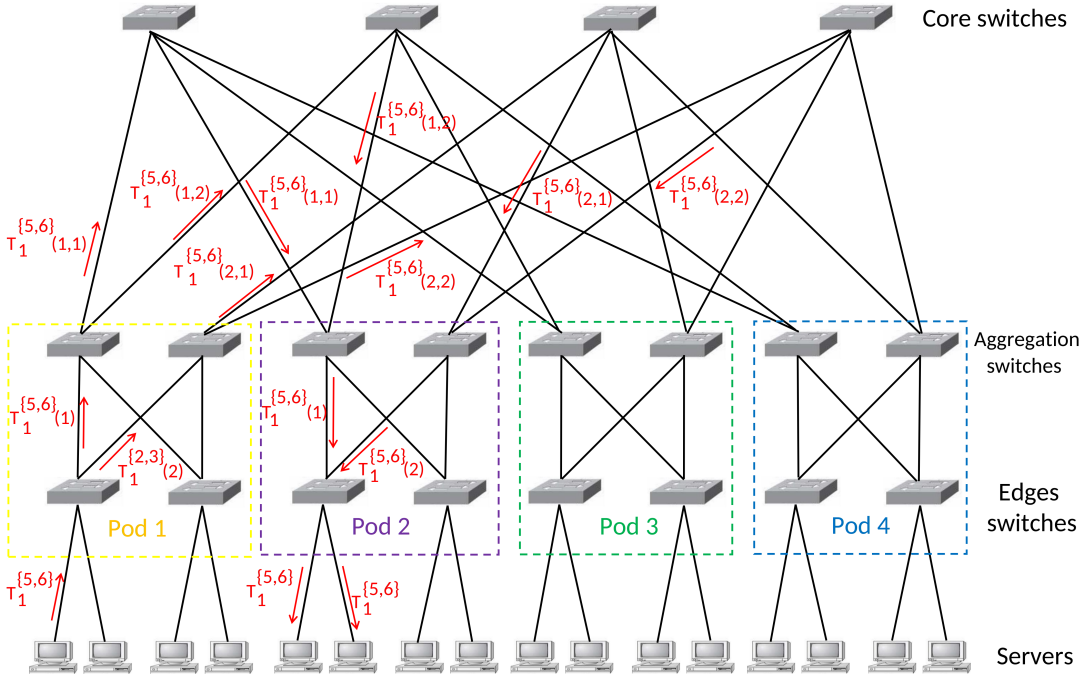

Fig. 2: The 4-ary fat tree, with $\frac{5t^2}{4}$ switches in total laying in the top three layers and $\frac{t^3}{4}$ servers laying in the bottom layer. Each of the $\frac{t^2}{4}$ core switches is connected to t aggregation switches. Each of the $\frac{t^2}{2}$ aggregation switches is connected to $\frac{t}{2}$ core switches and to $\frac{t}{2}$ edge switches. Each of the $\frac{t^2}{2}$ edge switches is connected to $\frac{t}{2}$ aggregation switches and to $\frac{t}{2}$ servers. The red parts represent the transmission of $T_1^{\{5,6\}}$ from server 1 to servers 5 and 6 through the network.

K. There are four layers in the topology, with $5t^2/4$ switches in total laying in the top three layers and $\frac{t^3}{4}$ servers laying in the bottom layer. The switches in the top three layers are referred to as *core switches*, *aggregation switches*, and *edge switches*, respectively, where the numbers of core switches, aggregation switches, and edge switches are $\frac{t^2}{4}$, $\frac{t^2}{2}$, and $\frac{t^2}{2}$, respectively.

The $\frac{t^2}{4}$ core switches are denoted by $\mathsf{c}_1, \ldots, \mathsf{c}_{\frac{t^2}{4}}$ from left to right in the network. A t-ary fat-tree topology contains t pods. We focus on pod i where $i \in [t]$. Pod i contains $\frac{t}{2}$ aggregation switches (denoted by $\mathsf{a}_{i,1}, \ldots, \mathsf{a}_{i,\frac{t}{2}}$ from the LHS to the RHS) and $\frac{t}{2}$ edge switches (denoted by $\mathsf{e}_{i,1}, \ldots, \mathsf{e}_{i,\frac{t}{2}}$ from left to right). Each aggregation switch $\mathsf{a}_{i,j}$ where $j \in \left[\frac{t}{2}\right]$ is connected to $\frac{t}{2}$ different core switches (core switches $\mathsf{c}_{\frac{(j-1)t}{2}+1}, \ldots, \mathsf{c}_{\frac{jt}{2}}$), such that each core switch is connected to exactly one aggregation switch in this pod. Aggregation switch $\mathsf{a}_{i,j}$ is also connected to each edge switch in this pod. Furthermore, each edge switch $\mathsf{e}_{i,p}$ where $p \in \left[\frac{t}{2}\right]$ is connected to $\frac{t}{2}$ servers at the bottom, and the positions of these servers are denoted by $\mathsf{w}_{i,p,1}, \ldots, \mathsf{w}_{i,p,\frac{t}{2}}$.

Hence, each switch in the fat-tree has t ports, such that $\frac{5t^2}{4}$ t-ports switches can handle up to $\frac{t^3}{4}$ servers. The cost to build this network is much cheaper than one $\frac{t^3}{4}$-ports switch.[2]

C. Coded Distributed Computing through t-ary Fat-tree

Next, we will show that with this low-cost and scalable topology, there exists an achievable scheme which can also achieve the optimal communication load in (4). The proposed achievable scheme is also based on the coded distributed computing scheme in [9]. The map and reduce phases are the same as the scheme in [9]. In the following, we will describe how to deliver the messages $\{T_k : k \in [\mathsf{K}]\}$ through the t-ary fat-tree. The main intuition why the t-ary Fat-tree can lead to the optimal communication load is that **each edge switch or aggregation switch is connected to $\frac{t}{2}$ switches/servers at its lower layer and connected to $\frac{t}{2}$ switches at its higher layer, such that the load on each outgoing link of one switch can be no more than each of its incoming links.**

Based on the number of servers K, we choose

$$t = \min\left\{t_1 \in \mathbb{Z} : \frac{t_1^3}{4} \geq \mathsf{K}\right\}.$$

If $\mathsf{K} < \frac{t^3}{4}$, we place the K servers in the first K positions from the left at the bottom. For each $i \in [t]$, $p \in \left[\frac{t}{2}\right]$, and $s \in \left[\frac{t}{2}\right]$, if

[2]The cost of cables/links is much lower than the cost of switches. Hence, as in [17], in this paper we do not consider the cost of cables.

Authorized licensed use limited to: The University of Utah. Downloaded on October 01,2021 at 16:54:17 UTC from IEEE Xplore. Restrictions apply.

there is one server (assumed to be server k) placed in position $\mathsf{w}_{i,p,s}$, with a slight abuse of notation, we let $\mathsf{w}_{i,p,s} = k$; otherwise, $\mathsf{w}_{i,p,s} = 0$. In addition, we define that $\mathcal{M}_0 = \mathcal{W}_0 = T_0 = \emptyset$.

Uplink transmission for pod $i \in [t]$.

- Each server $\mathsf{w}_{i,p,s}$ where $p \in \left[\frac{t}{2}\right]$ and $s \in \left[\frac{t}{2}\right]$, sends $T_{\mathsf{w}_{i,p,s}}$ to its connected edge switch $\mathsf{e}_{i,p}$.
  The number of bits transmitted through the link from server $\mathsf{w}_{i,p,s}$ to edge switch $\mathsf{e}_{i,p}$ is
$$|T_{\mathsf{w}_{i,p,s}}| \leq L^\star(r)\frac{\mathsf{QNT}}{\mathsf{K}}. \tag{8}$$
- We focus on edge switch $\mathsf{e}_{i,p}$ where $p \in \left[\frac{t}{2}\right]$. For each $s \in \left[\frac{t}{2}\right]$, edge switch $\mathsf{e}_{i,p}$ divides $T_{\mathsf{w}_{i,p,s}}$ into $\frac{t}{2}$ non-overlapping and equal-length pieces, denoted by $T_{\mathsf{w}_{i,p,s}}(1), \ldots, T_{\mathsf{w}_{i,p,s}}(\frac{t}{2})$. Recall that $T_k^{\mathcal{S}}$ represents the sub-message in T_k which are exclusively useful to servers in $\mathcal{S}$. For each $j \in \left[\frac{t}{2}\right]$, we define $T^{\mathcal{S}}_{\mathsf{w}_{i,p,s}}(j)$ as the set of bits in $T_{\mathsf{w}_{i,p,s}}(j)$ which are exclusively useful to servers in $\mathcal{S}$. The above partition of $T_{\mathsf{w}_{i,p,s}}$ is symmetric, i.e.,
$$\left|T^{\mathcal{S}}_{\mathsf{w}_{i,p,s}}(1)\right| = \cdots = \left|T^{\mathcal{S}}_{\mathsf{w}_{i,p,s}}(\frac{t}{2})\right| = \frac{2|T^{\mathcal{S}}_{\mathsf{w}_{i,p,s}}|}{t},$$
for each $\mathcal{S} \subseteq [\mathsf{K}] \setminus \{\mathsf{w}_{i,p,s}\}$. Edge switch $\mathsf{e}_{i,p}$ then sends $T_{\mathsf{w}_{i,p,s}}(j)$ to aggregation switch $\mathsf{a}_{i,j}$ for each $j \in \left[\frac{t}{2}\right]$.
  The number of bits transmitted through the link from edge switch $\mathsf{e}_{i,p}$ to aggregation switch $\mathsf{a}_{i,j}$ is
$$\begin{aligned}\sum_{s\in\left[\frac{t}{2}\right]} |T_{\mathsf{w}_{i,p,s}}(j)| &= \sum_{s\in\left[\frac{t}{2}\right]} \frac{2|T_{\mathsf{w}_{i,p,s}}|}{t} \\ &\leq \frac{L^\star(r)\mathsf{QNT}}{\mathsf{K}}.\end{aligned} \tag{9}$$
- We then focus on aggregation switch $\mathsf{a}_{i,j}$ where $j \in \left[\frac{t}{2}\right]$. For each $p \in \left[\frac{t}{2}\right]$ and $s \in \left[\frac{t}{2}\right]$, aggregation switch $\mathsf{a}_{i,j}$ further divides $T_{\mathsf{w}_{i,p,s}}(j)$ into $\frac{t}{2}$ non-overlapping and equal-length pieces, denoted by $T_{\mathsf{w}_{i,p,s}}(j,1), \ldots, T_{\mathsf{w}_{i,p,s}}(j,\frac{t}{2})$. For each $d \in \left[\frac{t}{2}\right]$, We also define $T^{\mathcal{S}}_{\mathsf{w}_{i,p,s}}(j,d)$ as the set of bits in $T_{\mathsf{w}_{i,p,s}}(j,d)$ which are exclusively useful to servers in $\mathcal{S}$. The above partition of $T_{\mathsf{w}_{i,p,s}}(j)$ is also symmetric, such that
$$\left|T^{\mathcal{S}}_{\mathsf{w}_{i,p,s}}(j,1)\right| = \cdots = \left|T^{\mathcal{S}}_{\mathsf{w}_{i,p,s}}(j,\frac{t}{2})\right| = \frac{2|T^{\mathcal{S}}_{\mathsf{w}_{i,p,s}}(j)|}{t},$$
for each $\mathcal{S} \subseteq [\mathsf{K}]\setminus\{\mathsf{w}_{i,p,s}\}$. Aggregation switch $\mathsf{a}_{i,j}$ sends $T_{\mathsf{w}_{i,p,s}}(j,d)$ to core switch $\mathsf{c}_{\frac{(j-1)t}{2}+d}$ for each $d \in \left[\frac{t}{2}\right]$.
  The number of bits transmitted through the link from aggregation switch $\mathsf{a}_{i,j}$ to core switch $\mathsf{c}_{\frac{(j-1)t}{2}+d}$ is
$$\begin{aligned}&\sum_{p\in\left[\frac{t}{2}\right]}\sum_{s\in\left[\frac{t}{2}\right]} |T_{\mathsf{w}_{i,p,s}}(j,d)| = \sum_{p\in\left[\frac{t}{2}\right]}\sum_{s\in\left[\frac{t}{2}\right]} \frac{2|T_{\mathsf{w}_{i,p,s}}(j)|}{t} \\ &= \sum_{p\in\left[\frac{t}{2}\right]}\sum_{s\in\left[\frac{t}{2}\right]} \frac{4|T_{\mathsf{w}_{i,p,s}}|}{t^2} \\ &\leq \frac{L^\star(r)\mathsf{QNT}}{\mathsf{K}}.\end{aligned} \tag{10}$$

Before introducing the downlink transmission, for each pod $i \in [t]$, we define $\mathcal{N}_i = \{\mathsf{w}_{i,p,s} : p \in \left[\frac{t}{2}\right], s \in \left[\frac{t}{2}\right]\}$, as the set of servers connected to the edge switches in pod i.

Downlink transmission for pod $i \in [t]$.

- We focus on aggregation switch $\mathsf{a}_{i,j}$ where $j \in \left[\frac{t}{2}\right]$. For each $d \in \left[\frac{t}{2}\right]$, core switch $\mathsf{c}_{\frac{(j-1)t}{2}+d}$ sends to aggregation switch $\mathsf{a}_{i,j}$,
$$\left\{T_k^{\mathcal{S}}(j,d) : k \in [\mathsf{K}] \setminus \mathcal{N}_i, \mathcal{S} \subseteq [\mathsf{K}] \setminus \{k\}, \mathcal{S} \cap \mathcal{N}_i \neq \emptyset\right\}.$$
  Notice that the aggregation switches in pod i have already received the bits in T_k for $k \in \mathcal{N}_i$ from the edge switches in this pod, and thus the aggregation switches need not to receive those bits from the core switches.
  The number of bits transmitted through the link from core switch $\mathsf{c}_{\frac{(j-1)t}{2}+d}$ to aggregation switch $\mathsf{a}_{i,j}$ is no more than
$$\begin{aligned}&\sum_{u\in\mathcal{N}_i} \left|\left\{T_k^{\mathcal{S}}(j,d) : k \in [\mathsf{K}] \setminus \{u\}, \mathcal{S} \subseteq [\mathsf{K}] \setminus \{k\}, u \in \mathcal{S}\right\}\right| \\ &= \sum_{u\in\mathcal{N}_i} \frac{4}{t^2}\left|\left\{T_k^{\mathcal{S}} : k \in [\mathsf{K}] \setminus \{u\}, \mathcal{S} \subseteq [\mathsf{K}] \setminus \{k\}, u \in \mathcal{S}\right\}\right| \\ &\leq \left(\mathsf{N} - \frac{r\mathsf{N}}{\mathsf{K}}\right)\frac{\mathsf{QsT}}{\mathsf{K}}.\end{aligned} \tag{11}$$
- We then focus on edge switch $\mathsf{e}_{i,p}$ where $p \in \left[\frac{t}{2}\right]$. For each $j \in \left[\frac{t}{2}\right]$, the messages from aggregation switch $\mathsf{a}_{i,j}$ to edge switch $\mathsf{e}_{i,p}$ are given by
$$\begin{aligned}&\left\{T_k^{\mathcal{S}}(j) : k \in [\mathsf{K}] \setminus \{\mathsf{w}_{i,p,1}, \ldots, \mathsf{w}_{i,p,\frac{t}{2}}\}, \mathcal{S} \subseteq [\mathsf{K}] \setminus \{k\},\right. \\ &\left.\mathcal{S} \cap \{\mathsf{w}_{i,p,1}, \ldots, \mathsf{w}_{i,p,\frac{t}{2}}\} \neq \emptyset\right\}.\end{aligned} \tag{12}$$
  Notice that edge switch $\mathsf{e}_{i,p}$ have already received T_k for $k \in \{\mathsf{w}_{i,p,1}, \ldots, \mathsf{w}_{i,p,\frac{t}{2}}\}$ from its connected servers, and thus edge switch $\mathsf{e}_{i,p}$ needs not to receive those bits from the aggregation switches.
  The number of bits transmitted through the link from aggregation switch $\mathsf{a}_{i,j}$ to edge switch $\mathsf{e}_{i,p}$ is no more than
$$\begin{aligned}&\sum_{u\in\{\mathsf{w}_{i,p,1},\ldots,\mathsf{w}_{i,p,\frac{t}{2}}\}} \left|\left\{T_k^{\mathcal{S}}(j) : k \in [\mathsf{K}] \setminus \{u\},\right.\right. \\ &\qquad\left.\left.\mathcal{S} \subseteq [\mathsf{K}] \setminus \{k\}, u \in \mathcal{S}\right\}\right| \\ &= \sum_{u\in\{\mathsf{w}_{i,p,1},\ldots,\mathsf{w}_{i,p,\frac{t}{2}}\}} \frac{2}{t}\left|\left\{T_k^{\mathcal{S}} : k \in [\mathsf{K}] \setminus \{u\},\right.\right. \\ &\qquad\left.\left.\mathcal{S} \subseteq [\mathsf{K}] \setminus \{k\}, u \in \mathcal{S}\right\}\right| \\ &\leq \left(\mathsf{N} - \frac{r\mathsf{N}}{\mathsf{K}}\right)\frac{\mathsf{QsT}}{\mathsf{K}}.\end{aligned} \tag{13}$$
- Finally we focus on server $\mathsf{w}_{i,p,s}$ where $p \in \left[\frac{t}{2}\right]$ and $s \in \left[\frac{t}{2}\right]$. Edge switch $\mathsf{e}_{i,p}$ sends to server $\mathsf{w}_{i,p,s}$,
$$\begin{aligned}U_{\mathsf{w}_{i,p,s}} = \Big\{&T_k^{\mathcal{S}} : k \in [\mathsf{K}] \setminus \{\mathsf{w}_{i,p,s}\},\\ &\mathcal{S} \subseteq [\mathsf{K}] \setminus \{k\}, \mathsf{w}_{i,p,s} \in \mathcal{S}\Big\}.\end{aligned}$$

Authorized licensed use limited to: The University of Utah. Downloaded on October 01,2021 at 16:54:17 UTC from IEEE Xplore. Restrictions apply.

The number of bits transmitted through the link from edge switch $\mathsf{e}_{i,p}$ to server $\mathsf{w}_{i,p,s}$ is no more than

$$\left(\mathsf{N}-\frac{r\mathsf{N}}{\mathsf{K}}\right)\frac{\mathsf{QsT}}{\mathsf{K}}. \tag{14}$$

By summing (8) and (11), summing (9) and (13), summing (10) and (14), it can be seen that the total number of bits transmitted through each link in the t-ary fat-tree is no more than

$$\frac{L^{\star}(r)\mathsf{QNT}}{\mathsf{K}}+\left(\mathsf{N}-\frac{r\mathsf{N}}{\mathsf{K}}\right)\frac{\mathsf{QsT}}{\mathsf{K}}.$$

Hence, we prove that the proposed scheme through the t-ary fat-tree can achieve the optimal communication load in (4).

To explain explicitly our proposed scheme through the t-ary fat-tree network, in Fig. 2 we illustrate the transmission of $T_1^{\{5,6\}}$ (which is sent by server 1 and useful to servers 5 and 6).

Remark 1. *In the proposed distributed computing scheme, the switches only receive and forward packets. However, due to the congestion over the links, while receiving one packet, a switch may not be possible to forward it immediately. Hence, the switch needs a buffer to put this packet in the transmission queue. As a matter of fact, in real systems switches always are equipped with buffers.*

D. Discussions

In this section, we will discuss two other practical issues.

Congestion at higher links: In hierarchical computing networks, the traffic at the top layers is always higher than the bottom layers and the higher links always need to have higher capacities to avoid the congestion. However, in our t-ary fat-tree coded computing system, as we proved in Section III-C, all links in the fat-tree topology have similar link loads (the traffic at the bottom layers is slightly higher than the top layers) and thus we can build the network using links with the same capacity, which also reduces the construction cost.

Fault-tolerance: On the one hand, any failure happening at the switches or the links among the switches can be tolerated, because there are multiple paths from each server to another (each edge switch is connected $\frac{t}{2}$ aggregation switches and each aggregation switch is connected to $\frac{t}{2}$ core switches). On the other hand, if a server node or the link connected to it fails, we can add some redundancies in the system by using some form of the Minimum Distance Separable (MDS) code as used in [7]. However, the detailed description of such schemes is beyond the scope of this paper.

IV. Conclusions

In this paper, we considered the topological coded distributed computing problem. We first characterized the optimal max-link communication load over any network topology and showed that it could be achieved by using the single-switch topology. To reduce the cost to build the network and enable the scalability, we then considered the t-ary fat-tree topology. We then proposed a coded distributed computing scheme through the t-ary fat tree, which can indeed achieve the optimal max-link communication load. The proposed scheme can avoid the congestion at higher links and tolerate the failures in the network components, such that our result has both a significant intellectual merit and a high practical relevance. Finally, the proposed scheme can be easily extended to other computing problems with distributed servers, such as decentralized data shuffling [19].

Acknowledgements

This work has been partially funded by the ERC Advanced Grant Grant No. 789190 "CARENET", and the National Science Foundation grants CCF-1817154 and SpecEES-1824558.

References

[1] E. Amazon, "Amazon web services," *Available in: http://aws. amazon. com/es/ec2/(November 2012)*, 2015.

[2] K. S. P. T. and L. U. Gonzalez, *Building Your Next Big Thing with Google Cloud Platform: A Guide for Developers and Enterprise Architects*. Apress, 2015.

[3] B. Wilder, *Cloud architecture patterns: using microsoft azure*. " O'Reilly Media, Inc.", 2012.

[4] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[5] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets." *HotCloud*, vol. 10, no. 10-10, p. 95, 2010.

[6] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Infor. Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.

[7] M. Ji, G. Caire, and A. Molisch, "Fundamental limits of caching in wireless d2d networks," *IEEE Trans. Inf. Theory*, vol. 62, no. 1, pp. 849–869, 2016.

[8] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Coded mapreduce," in *Communication, Control, and Computing (Allerton), 2015 53rd Annual Allerton Conference on*. IEEE, 2015, pp. 964–971.

[9] 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.

[10] N. Woolsey, R. Chen, and M. Ji, "A new combinatorial design of coded distributed computing," in *2018 IEEE International Symposium on Information Theory (ISIT)*, June 2018, pp. 726–730.

[11] K. Konstantinidis and A. Ramamoorthy, "Resolvable designs for speeding up distributed computing," 2019.

[12] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Compressed coded distributed computing," in *2018 IEEE International Symposium on Information Theory (ISIT)*, June 2018, pp. 2032–2036.

[13] S. R. Srinivasavaradhan, L. Song, and C. Fragouli, "Distributed computing trade-offs with random connectivity," in *2018 IEEE International Symposium on Information Theory (ISIT)*, June 2018, pp. 1281–1285.

[14] Y. H. Ezzeldin, M. Karmoose, and C. Fragouli, "Communication vs distributed computation: An alternative trade-off curve," in *2017 IEEE Information Theory Workshop (ITW)*, Nov 2017, pp. 279–283.

[15] S. Li, Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "A scalable framework for wireless distributed computing," *IEEE/ACM Transactions on Networking*, vol. 25, pp. 2643 – 2654, Oct. 2017.

[16] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Coding for distributed fog computing," *IEEE Communications Magazine*, vol. 55, pp. 34–40, Apr. 2017.

[17] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 63–74, 2008.

[18] W. Xia, P. Zhao, and Y. Wen, "A survey on data center networking (dcn): Infrastructure and operations," *IEEE Communications Surveys and Tutorials*, vol. 19, pp. 640 – 656, 2017.

[19] K. Wan, D. Tuninetti, M. Ji, G. Caire, and P. Piantanida, "Fundamental limits of decentralized data shuffling," *IEEE Transactions on Information Theory, doi: 10.1109/TIT.2020.2966197*, Jan. 2020.

Authorized licensed use limited to: The University of Utah. Downloaded on October 01,2021 at 16:54:17 UTC from IEEE Xplore. Restrictions apply.