Coded Computation over Heterogeneous Workers with Random Task Arrivals

Fan Zhang, Yuxuan Sun, *Member, IEEE,* Sheng Zhou, *Member, IEEE*

***Abstract*—Considering the scheduling and allocation of tasks among multiple servers, distributed machine learning faces the problem of the straggler effect as well as system heterogeneity, e.g., the computation time of the slowest worker can be much longer than that of the normal workers. This letter studies the distributed online tasks assignment problem under heterogeneous conditions where different workers have different computing capacities, in order to minimize the task completion time. We consider the task scheduling with random task arrivals, and introduce task cancellation after completion scheme to clear the unfinished parts after the completion of the task to further reduce redundant calculations. To address the challenging of finding the optimal solution, we propose an approximate online algorithm based on convex optimization and time recursion. Simulation results show that the proposed algorithm can reduce the completion delay by over 30% as compared with the one-shot counterpart, and maintain a relatively stable delay in the case of fluctuating arrival rates.**

I. Introduction

The emerging of big data and the penetration of numerous edge services motivate distributed machine learning, where a centralized node, called *master*, can pass computational data to distributed *workers*, and then collect their results to recover the overall computation result. This distributed architecture allows large amounts of data to be processed at the same time by parallelization, leading to substantial reduction of the computing time.

The existence of *straggler effect* greatly deteriorates the performance of distributed computing system, due to the heterogeneous computing capabilities of workers and the effect of the slowest worker. Experiments on Amazon Elastic Compute Cloud show that the speed of some workers may be 5 times slower than that of other normal workers [1]. This problem can be addressed by coding to impose redundancy, which allows a master to recover results from a subset of workers, and by optimizing task assignment algorithms to balance the task completion time among heterogeneous workers.

There are different encoding schemes proposed in [1]–[9], which are applicable to different conditions. For example, maximum distance separable (MDS) codes are widely applied for matrix multiplications [3], [6]–[8], while gradient codes are used for updating the gradients of machine learning model [1]. The coding scheme based on discrete avoidance with MDS coding is proposed in [4]. In [5], the authors consider the multi-level computing network model with tree-like coding structure. Paper [7] investigates the age performance of uncoded and coded schemes including repetition coding, MDS coding, and multi-message MDS coding. The random gradient coding proposed in [8] improves the performance when stragglers are random and abundant. In the case of large data set and limited computing power, a single-communication-round coding scheme is proposed in [9] to tune the amount of computing workloads for each woker in advance.

Different scheduling algorithms are proposed in [6], [10]–[13]. Workers ranked by their straggling levels are assigned with different sizes of tasks to exploit the computing power of stragglers in [6]. In [10], the authors propose the deterministic scheduling orders for different workers, namely cyclic scheduling and staircase scheduling schemes. The assignment of multiple tasks of multiple masters is further considered in [11]. A task scheduling algorithm considering task priority is proposed in [12]. The expected computing time of the distributed algorithm is analyzed in [13].

However, these works mainly consider the one-shot task assignment problem i.e., there is only a single task to be processed, or there are multiple tasks arriving at the same time. Since distributed machine learning needs to go through multiple rounds of training and data transmission, and different tasks may co-exist in the system, it is insufficient to consider the one-shot case. In this work, we focus on the delay-minimized online assignment of matrix multiplication tasks randomly arriving at the master, with the following key contributions: 1) We extend the state-of-the-art one-shot coded task assignment scenario to a more general case with random task arrivals, and incorporate the task cancellation strategy to further improve the system efficiency. 2) We propose an online task assignment algorithm by recursive start time approximation and optimal load allocation, which can be applied to any task arrival process. 3) Simulations under a time-varying task arrival rate scenario validate that, the proposed algorithm can achieve lower task completion delay compared to all benchmarks, and is robust to different levels of system heterogeneity.

II. System Model and Problem Formulation

A. Task Encoding

As shown in Fig. 1, we consider a heterogeneous distributed computing system, where a master can send matrix-multiplication tasks to N workers $\mathcal{N} = \{1, 2, ..., N\}$ with

F. Zhang, Y. Sun and S. Zhou are with Beijing National Research Center for Information Science and Technology, Department of Electronic Engineering, Tsinghua University, Beijing 10084, China. Emails: {zhang-f17@mails., sunyuxuan@, sheng.zhou@}tsinghua.edu.cn.

This work is sponsored in part by the National Key R&D Program of China 2018YFB1800804, in part by the Nature Science Foundation of China (No. 62022049, No. 61871254, No. 91638204, No. 61861136003), in part by the China Postdoctoral Science Foundation No. 2020M680558, and Hitachi Ltd.

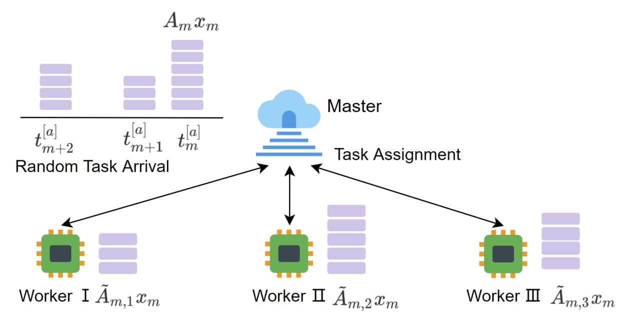

Fig. 1. Illustration of a distributed computing system with random task arrivals.

different computing capabilities. The m-th task is defined as $\boldsymbol{A}_m\boldsymbol{x}_m$, where matrix $\boldsymbol{A}_m \in \mathbb{R}^{L_m\times W_m}$, vector $\boldsymbol{x}_m \in \mathbb{R}^{W_m}$, $L_m, W_m \in \mathbb{Z}^+$, and $\mathbb{Z}^+$ represents the positive integer set. The arrival time of task m is denoted by $t^{[a]}_m$.

In order to reduce the impact of stragglers, the master encodes the rows of $\boldsymbol{A}_m$ through MDS coding. Define the encoded matrix as $\tilde{\boldsymbol{A}}_m$. Then the master assigns each worker an encoded task, which is represented as $\tilde{\boldsymbol{A}}_{m,n} \in \mathbb{R}^{l_{m,n}\times W_m}$, $\tilde{\boldsymbol{A}}_m = \left[\tilde{\boldsymbol{A}}^T_{m,1},\ \tilde{\boldsymbol{A}}^T_{m,2}, \cdots, \tilde{\boldsymbol{A}}^T_{m,N}\right]^T$ with $W_m, l_{m,n} \in \mathbb{Z}^+$ and $\sum_{n=1}^N l_{m,n} \geq L_m$. The subscript (m, n) indicates that it is the part of the m-th task assigned to the n-th worker, and $l_{m,n}$ represents the corresponding workload. In view of the MDS coding, we can recover the result of the m-th task after collecting any L_m lines of computation from workers.

B. Queueing and Cancellation

At each worker, tasks are processed according to the first-come-first-serve discipline. Upon the arrival of the m-th task, the index of the unfinished task with earliest arrival time in the system is denoted by k, with $k \leq m$. Let $\boldsymbol{Q}_n(t) = \{q_{k,n}(t), ..., q_{m-1,n}(t), q_{m,n}(t)\}$ be the task queue of the n-th worker at time t. The cardinality of $\boldsymbol{Q}_n(t)$ represents the number of unfinished tasks in the system, while each element $q_{j,n}(t), j = k, \cdots, m$ represents the remaining load of task j at worker n. It is easy to see that $q_{k,n}(t) \leq l_{k,n}$ because rows in the queue are automatically removed after being calculated, where the inequality holds only for the task at the head of the queue.

Let $t^{[a]-}_m$ and $t^{[a]+}_m$ be the time instances right before and after the arrival of task m at the master. In this work, we mainly focus on the computation bottleneck, and assume that the delay consumed by running the task assignment algorithm and transmitting the data of coded subtasks can be ignored. Therefore, $t^{[a]+}_m$ also indicates the arrival time of encoded subtask $\tilde{\boldsymbol{A}}_{m,n}\boldsymbol{x}_m$ at any worker $n \in \mathcal{N}$, and

$$\boldsymbol{Q}_n\left(t^{[a]+}_m\right) = \boldsymbol{Q}_n\left(t^{[a]-}_m\right) \bigcup \{l_{m,n}\} = \{q_{k,n}(t), l_{k+1,n}, \cdots, l_{m,n}\}.$$

To simplify the system, we assume that for each task, the worker will not transmit the result back to the master until it finishes the subtask [11]. Therefore, the master will either receive $l_{m,n}$ results (one result refers to one row of the multiplication of $\tilde{A}_m$ and x_m) or receive nothing from worker n at time t, which is denoted as $R_{m,n}(t)$. We consider *cancellation-upon-completion* to remove the unnecessary computations, to be specific, once the master recovers the result of the m-th task, the unfinished coded subtasks of it are cancelled with negligible delay. Let $t^{[e]}_k$ and $\hat{t}^{[e]}_{k,n}$ be the completion time of the whole k-th task and that at worker n, respectively. If $t^{[e]}_k < \hat{t}^{[e]}_{k,n}$, then the remaining load in the queue of worker n are cleared:

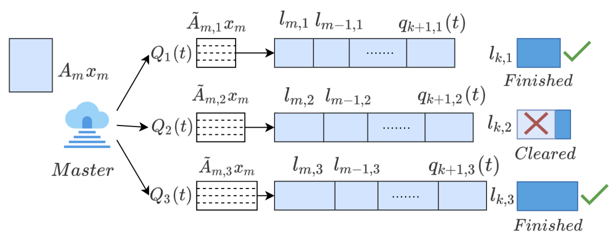

Fig. 2. Queue update.

$$\boldsymbol{Q}_n\left(t^{[e]+}_k\right) = \boldsymbol{Q}_n\left(t^{[e]-}_k\right) \backslash \{q_{k,n}\left(t^{[e]-}_k\right)\} = \{l_{k+1,n}, \cdots, l_{m,n}\}.$$

If task m is not the first one in the queue of worker n, then the remaining task is $l_{k,n}$, that is, the $q_{k,n}(t^{[e]-}_k)$ should be replaced by $l_{k,n}$ in the equation above. An illustration of the queue updates is shown in Fig. 2.

C. Task Processing Delay

Referring to the model in [3], [6], [11], we express the processing time of $l_{m,n}$ results at worker n, denoted by $T_{l_{m,n}}$, as a random variable with shifted exponential distribution:

$$\mathbb{P}\left[T_{l_{m,n}} \leq t\right] = \begin{cases} 1 - e^{-\frac{u_n}{l_{m,n}}(t - a_n l_{m,n})}, & t \geq a_n l_{m,n}, \\ 0, \quad \text{otherwise}, \end{cases} \tag{1}$$

where $a_n > 0$ is a parameter indicating the minimum processing time for one coded row and $u_n > 0$ is the parameter modeling the straggling effect. Different a_n and u_n represent different computing capabilities.

D. Problem Formulation

Our objective is to minimize the completion time $t^{[e]}_m$ of task m, upon which the master can get sufficient results from workers to recover the computation. We aim to design a centralized policy to optimize the load allocation $\{l_{m,n}\}$ upon the arrival of each task. Comparing to the single-task assignment policy, we further incorporate the queue state of each worker for online scheduling. Therefore, the problem is formulated as minimizing the completion time $t^{[e]}_m$ of current task with the information of current queue state $\boldsymbol{Q}(t^{[a]}_m)$:

$$\mathcal{P}1: \min_{\{l_{m,n}\}} \quad \mathbb{E}\left[t^{[e]}_m | \boldsymbol{Q}(t^{[a]}_m)\right] \tag{2a}$$

$$\text{s.t.} \quad \mathbb{P}\left[\sum_n R_{m,n}(t^{[e]}_m) \geq L_m\right] \geq \sigma. \tag{2b}$$

In constraint (2b), σ is defined as the probability that the master receives at least L_m results, meaning that task m is finished, until time $t^{[e]}_m$. The main difficulty to solve $\mathcal{P}1$ is that, constraint (2b) is hard to express mathematically as the combination of returned tasks satisfying $\sum_n R_{m,n}(t^{[e]}_m) \geq L_m$

is difficult to be explicitly found in a heterogeneous condition. Define $\mathbb{E}\Big[R_m(t_m^{[e]})\Big]=\mathbb{E}\Big[\sum_n R_{m,n}(t_m^{[e]})\Big]$. To solve this problem, we replace the probability with the expectation as follow:

$$\mathcal{P}2: \min_{\{l_{m,n}\}} \quad \mathbb{E}\left[t_m^{[e]}|\boldsymbol{Q}(t_m^{[a]})\right] \tag{3a}$$

$$\text{s.t.} \quad L_m - \mathbb{E}\left[R_m(t_m^{[e]})\right] \le 0. \tag{3b}$$

According to [14], the gap between the solutions of $\mathcal{P}1$ and $\mathcal{P}2$ can be bounded with a single master. The key challenge to solve $\mathcal{P}2$ is that all the unfinished parts will be cleared when the task is completed, so it is still arduous to get the arithmetic expression of $\mathbb{E}[R_m(t_m^{[e]})]$.

Observe that if we get the calculation start time of each subtask at each worker, the challenge mentioned above can be solved. Define the time starting the computation of task m at worker n as $t_{m,n}^{[s]}$ which is expressed as the larger value between the previous task completion time $t_{m-1,n}^{[e]}$ and the arrival time $t_m^{[a]}$. Define $\boldsymbol{t}_m^{[s]}=\{t_{m,1}^{[s]}, t_{m,2}^{[s]}...t_{m,N}^{[s]}\}$. The length of $\boldsymbol{Q}(t_m^{[a]})$ provides information to calculate the theoretical completion time $t_{m-1,n}^{[e]}$, for all $n \in \mathcal{N}$. Thus, the problem can be divided into two parts, namely using $\boldsymbol{Q}(t_m^{[a]})$ to obtain $t_{m-1,n}^{[s]}$ and getting $t_m^{[e]}$ with $t_{m-1,n}^{[s]}$. Using the expectation formula, we can convert $\mathcal{P}2$ into two levels of expectation:

$$\mathcal{P}3: \min_{\{l_{m,n}\}} \quad \mathbb{E}_{\boldsymbol{t}_m^{[s]}}\left[\mathbb{E}_{t_m^{[e]}|\boldsymbol{t}_m^{[s]}}\left[t_m^{[e]}|\boldsymbol{t}_m^{[s]}\right]\right] \tag{4a}$$

$$\text{s.t.} \quad L_m - \mathbb{E}_{\boldsymbol{t}_m^{[s]}}\left[\mathbb{E}_{R_m(t_m^{[e]})|\boldsymbol{t}_m^{[s]}}\left[R_m(t_m^{[e]})|\boldsymbol{t}_m^{[s]}\right]\right] \le 0. \tag{4b}$$

Define the density function of variable $t_{m,n}^{[s]}$ as $f_{m,n}^s(t)$, and $f_m^s(t)=\prod_{n=1}^N f_{m,n}^s(t)$. In this condition, $\mathbb{E}[R_m(t_m^{[e]})]$ can be expressed with the given $\boldsymbol{t}_m^{[s]}$:

$$\mathbb{E}_{\boldsymbol{t}_m^{[s]}}\left[\mathbb{E}_{R_m(t_m^{[e]})|\boldsymbol{t}_m^{[s]}}\left[R_m(t_m^{[e]})|\boldsymbol{t}_m^{[s]}\right]\right] = \int_{n=1}\!...\!\int_{n=N}$$
$$\left(\sum_{n=1}^N l_{m,n}\left[1-e^{-\frac{u_n}{l_{m,n}}\left(t_m^{[e]}-t_{m,n}^{[s]}-a_n l_{m,n}\right)}\right]\right) f_m^{[s]}(\boldsymbol{t}_m^{[s]})d\boldsymbol{t}_m^{[s]}. \tag{5}$$

Since the density function $f_{m,n}^s(t)$ depends on the density function $f_{m-1,n}^s(t)$ of the previous task, the recurrence relation expression can be obtained from distribution (1):

$$f_{m,n}^s(t)=$$
$$\begin{cases} \frac{u_n}{l_{m,n}}e^{(t-t_m^{[a]}-a_n l_{m,n})}\int_0^{t_m^{[a]}} f_{m-1,n}^s(t_0)dt_0, & t_{m-1,n}^{[e]} \le t_m^{[a]}, \\ \int_{t_m^{[a]}}^t f_{m-1,n}^s(t_0)\left(\frac{u_n}{l_{m,n}}e^{(t-t_0-a_n l_{m,n})}\right)dt_0, & t_{m-1,n}^{[e]} > t_m^{[a]}. \end{cases}$$

From the recursion relationship, to obtain the density function $f_{m,n}^s(t)$, we must calculate the completion time of the first task in the current queue. Define $\Omega_m=\{\omega_{m,1}..\omega_{m,r}\}$ as a collection in which $\sum_{n\in\omega_{m,p}} l_{m,n} \ge L_m$, for $p \in 1,...,r$. Ω_m represents the set containing all possible return combinations of workers to complete task m. The probability can be illustrated as:

$$\mathbb{P}_{t_1^{[e]}}(t)=\sum_{\omega\in\Omega_1}\prod_{k\in\omega}\mathbb{P}\left(t_{1,k}^{[e]} \le t\right). \tag{6}$$

Considering that the combination shown in (6) can hardly be expressed explicitly, the probability distribution of the completion time of the first task is difficult to solve, and thus brings difficulties in solving the probability distribution of the completion time of task $M-1$.

In the following section, we propose a recursive algorithm to approximate $\boldsymbol{t}_m^{[s]}$.

III. Task Assignment

In this section, we solve $\mathcal{P}3$ by dividing the problem into two parts. In the first part we propose the optimal allocation of current task given the start time of calculation, and in the second part we design a recursive algorithm to approximate the start time.

A. Assign Tasks with a Given Start Time

When $\boldsymbol{t}_m^{[s]}$ is given, problem $\mathcal{P}3$ can be expressed as:

$$\mathcal{P}4: \min_{\{l_{m,n}\}} \quad \mathbb{E}\left[t_m^{[e]}\right] \tag{7a}$$

$$\text{s.t.} \quad L_m - \sum_{n=1}^N l_{m,n}\Big[1-e^{-\frac{u_n}{l_{m,n}}\left(t_m^{[e]}-t_{m,n}^{[s]}-a_n l_{m,n}\right)}\Big] \le 0, \tag{7b}$$

which is a convex optimization problem.

Define $\alpha > 0$ as the Lagrange multiplier. The following solution can be obtained by using the Lagrange multiplier method and Karush-Kuhn-Tucker condition [11]:

$$\mathcal{L}(t_m^{[e]}, \alpha, \boldsymbol{t}_m^{[s]})=$$
$$t_m^{[e]}+\alpha\left(L_m-\sum_{n=1}^N l_{m,n}\left[1-e^{-\frac{u_n}{l_{m,n}}\left(t_m^{[e]}-t_{m,n}^{[s]}-a_n l_{m,n}\right)}\right]\right). \tag{8}$$

So problem $\mathcal{P}4$ can be transfered to:

$$\mathcal{P}5: \min_{\{l_{m,n}\}} \quad \mathcal{L}\left(t_m^{[e]}, \alpha, \boldsymbol{t}_m^{[s]}\right). \tag{9a}$$

Define $\mathcal{W}_{-1}(x)$ as the lower branch of Lambert W function, where $x \le -1$ and $\mathcal{W}_{-1}(xe^x)=x$. Let $\phi_{m,n} \triangleq \frac{1}{u_n}\left[-\mathcal{W}_{-1}(-e^{-u_n a_n-1})-1\right]$.

Theorem 1. *With a given start time $\boldsymbol{t}_m^{[s]}=\{t_{m,1}^{[s]}, t_{m,2}^{[s]}...t_{m,n}^{[s]}\}$, the optimal allocation $l_{m,n}^*$ and expected completion time $t_m^{[e]}$ are given by:*

$$t_m^{[e]}=\frac{L_m+\sum_{n=1}^N \frac{t_{m,n}^{[s]}u_n}{1+u_n\phi_{m,n}}}{\sum_{n=1}^N \frac{u_n}{1+u_n\phi_{m,n}}}, \tag{10}$$

$$l_{m,n}^*=\frac{t_m^{[e]}-t_{m,n}^{[s]}}{\phi_{m,n}}. \tag{11}$$

Here we approximate $l_{m,n}$ to a real number $l_{m,n}^*$.
Proof: See Appendix A.

B. Recursive Solution

We then design a recursive algorithm to approximate $\boldsymbol{t}_m^{[s]}$ in this subsection. According to the expression $t_{m,n}^{[s]}=\max\{t_{m-1,n}^{[e]}, t_m^{[a]}\}$, we need to calculate two parts, the completion time of task $m-1$ and the arrival time of task m respectively. Notice that the completion time of task $m-1$ of each worker can not be directly calculated, because it is

Algorithm 1 Recursive Start Time Approximation and Optimal Load Allocation

```
1:  Input: $\boldsymbol{u}=\{u_1,u_2,...u_N\}$, $\boldsymbol{a}=\{a_1,a_2,...a_N\}$, $\boldsymbol{Q}(t)=\{\boldsymbol{Q}_1(t),\boldsymbol{Q}_2(t)...\boldsymbol{Q}_N(t)\}$, $\boldsymbol{L}=\{L_k,L_2...L_M\}$, $\boldsymbol{t}^{[a]}=\{t_k^{[a]},t_{k+1}^{[a]}...,t_m^{[a]}\}$.
2:  for $m=k,...,M-1$ do
3:      Approximate $t_m^{[e]}$ with
4:      if $m==k$ then
5:          $L_m-\sum_{n=1}^{N} q_{m,n}\left[1-e^{-\frac{u_n}{q_{m,n}}\left(t_m^{[e]}-t_{m,n}^{[s]}-a_n q_{m,n}\right)}\right]=0.$
6:      else
7:          $L_m-\sum_{n=1}^{N} l_{m,n}\left[1-e^{-\frac{u_n}{l_{m,n}}\left(t_m^{[e]}-t_{m,n}^{[s]}-a_n l_{m,n}\right)}\right]=0.$
8:      end if
9:      for $n=1,...,N$ do
10:         if $m==k$ then
11:             $\mathbb{E}(\hat{t}_{m,n}^{[e]})=q_{m,n}(a_n+\frac{1}{u_n})+t_m^{[a]}.$
12:         else
13:             $\mathbb{E}(\hat{t}_{m,n}^{[e]})=l_{m,n}(a_n+\frac{1}{u_n})+t_{m,n}^{[s]}.$
14:         end if
15:         if $t_m^{[e]}>\mathbb{E}(\hat{t}_{m,n}^{[e]})$ then $t_{m,n}^{[e]}=\mathbb{E}(\hat{t}_{m,n}^{[e]})$.
16:         else $t_{m,n}^{[e]}=t_m^{[e]}$.
17:         end if
18:         if $t_{m,n}^{[e]}>t_{m+1}^{[a]}$ then $t_{m+1,n}^{[s]}=t_{m,n}^{[e]}$.
19:         else $t_{m+1,n}^{[s]}=t_{m+1}^{[a]}$.
20:         end if
21:     end for
22: end for
23: $t_M^{[e]}=\frac{L_M+\sum_{n=1}^{N}\frac{t_{M,n}^{[s]}u_n}{1+u_n\phi_{M,n}}}{\sum_{n=1}^{N}\frac{u_n}{1+u_n\phi_{M,n}}}.$
24: for $n=1,...,N$ do
25:     $l_{M,n}^{*}=\frac{t_M^{[e]}-t_{M,n}^{[s]}}{\phi_{M,n}}.$
26: end for
```

the minimum time between $\hat{t}_{m-1,n}^{[e]}$ which represents the time worker n completing task m–1 and the completion time of the whole task $t_{m-1}^{[e]}$. As shown in equation (6), it is hard to get the probability distribution of the completion time. To reduce the complexity of the algorithm and avoid unnecessary calculation burden, we use the result of $\mathbb{E}\left[\sum_{n=1}^{N}R_{m,n}(t_m^{[e]})\right]=L_m$ to approximate the completion time of the whole task $t_m^{[e]}$. With the distribution in (1) and the recursion relationship, we can obtain:

$$L_{m-1}-\sum_{n=1}^{N}l_{m-1,n}\left[1-e^{-\frac{u_n}{l_{m-1,n}}\left(t_{m-1}^{[e]}-t_{m-1,n}^{[s]}-a_n l_{m-1,n}\right)}\right]=0. \quad (12)$$

$$\mathbb{E}(\hat{t}_{m-1,n}^{[e]})=l_{m-1,n}\left(a_n+\frac{1}{u_n}\right)+t_{m-1,n}^{[s]}. \quad (13)$$

When the task $m-1$ is the first task in the queue, the remaining task load in the queue should be replaced by $q_{m-1,n}(t)$. Notice that (12) is a monotone function of $t_{m-1}^{[e]}$, and thus we can get the explicit value of $t_{m-1}^{[e]}$ under the given constraint $L_{m-1}\leq\sum_{n=1}^{N}l_{m-1,n}$.

Combining with the result in Theorem 1, the Recursive Start Time Approximation and Optimal Load Allocation are proposed in Algorithm 1. As shown in Lines 2-22, when the M-th task arrives at the master, the recursive approximation time for each task $m=k,\cdots,M-1$ is divided into two steps: 1) Use equations (12) and (13) to solve the completion time $t_{m,n}^{[e]}$, as shown in Lines 3-8. 2) Get the computation start time $t_{m+1,n}^{[s]}$ by comparing $t_{m,n}^{[e]}$ with the arrival time $t_{m+1}^{[a]}$, as shown in Lines 9-22. Finally, we calculate the optimal allocation $l_{M,n}^{*}$ for the current task M using Theorem 1, as shown in Lines 23-25. For each task m in the system, the complexity to calculate the start time is $O(N)$, and thus if there are $M-k$ tasks remaining in the system, the complexity of the algorithm is $O((M-k)N)$.

IV. Simulation Results

In this section, we evaluate the task completion time of the proposed Recursive Start Time Approximation and Optimal Load Allocation under different task arrival rates and compare the influence of computing capacity. Since the task arrival time of different algorithms is consistent, to be more intuitively, completion delay is shown instead of the completion time.

We consider $N=20$ workers with four benchmarks: 1) *Uncoded*: Divide each task equally, each worker gets $\frac{L_m}{N}$ tasks without coding. 2) *Mean Capability*: According to the expectation, each worker's computing capacity can be regarded as $\frac{1}{a+\frac{1}{u}}$. Assign tasks according to the computing proportion, and thus each worker gets $L_m\frac{\frac{u_n}{a_nu_n+1}}{\sum_n\frac{u_n}{a_nu_n+1}}$ rows. 3) *Queue-based*: Without clearing up unfinished parts, the expected completion time can be calculated directly from the queue length. Then allocate the tasks according to Theorem 1 given the expected completion time. 4) *One-shot assignment*: Assign tasks according to Theorem 1 regardless of queue length [11] and clear the unfinished parts.

The capacity parameter a_n is randomly selected within $[0.1,0.5]$ ms, the straggling parameter u_n is set as $u_n=\frac{5}{a_n}$ ms^{-1}, $n\leq16$ and $u_n=\frac{1}{5a_n}$ ms^{-1}, $17\leq n\leq20$. The length of the rows of multiplication task, L_m, is randomly between $[1700,2700]$ [14].

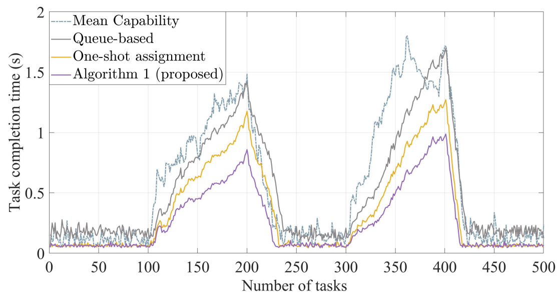

Fig. 3. Task completion delay under fluctuating arrival rates

We show the task completion delay of different algorithms in the case of varying arrival rates in Fig. 3. The average task arrival interval for tasks numbered [1-100,400-500] is 0.12s, for tasks numbered [100-200,300-400] is 0.04s, and for tasks numbered [200-300] is 0.08s. The uncoded algorithm is not used as a benchmark in this graph due to its large delay. Comparing the task completion delay with other benchmarks, the proposed algorithm is more stable and achieves lower delay. Without considering the queue state, mean capacity algorithm is the most unstable one. The gap between the

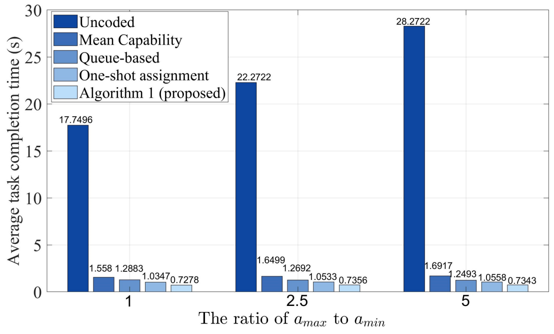

Fig. 4. The influence of computing capability distributions of workers on the average task completion delay.

queue-based algorithm and the proposed algorithm validates the task cancellation upon completion strategy. When the average arrival interval is 0.04s, the proposed algorithm can reduce the completion delay by over 60% as compared with the mean capacity algorithm, by around 40% over the queue-based algorithm and by 30% over the one-shot algorithm.

Fig. 4 shows the impact of difference in computing capabilities among workers. Keeping $\sum_n \frac{1}{a_n+\frac{1}{u_n}}$ and u_n unchanged which means the frequency of calculation and the stragglers' distribution remain the same, we use different distributions of a_n to simulate different computing capacities with the arrival of 0.4s. The x-axis of the Fig. 4 is the ratio of maximum and minimum computing capability representing heterogeneity among workers. We use a uniform frequency distribution of [1,5] of a_n as the reference frequency. Under the limitation of frequency and u_n, the value of a_n is 1.9175 when the ratio is 1 and the uniform distribution of [1.381,3.4525] when the ratio is 2.5. The average completion delay of 300 tasks shows that the computational difference between workers has large influence on uncoded and queue-blind algorithms while our algorithm is robust under different heterogeneous environments.

In summary, the proposed algorithm performs better than all the benchmarks and remains stable even with fluctuating arrival rates. If the queue state is not taken into account, such as mean-capacity algorithm, then its completion delay will be unstable. Meanwhile, comparing the proposed algorithm with the queue-based algorithm, the cancellation strategy helps maintain a lower delay by avoiding redundant computation.

V. Conclusion

We have considered an online task assignment problem in a heterogeneous coded computing system. MDS coding has been adopted to alleviate the straggler effect and the cancellation of the unfinished parts after completion of the whole task has been considered to further improve the efficiency. With the approximate start time, the task optimal allocation among workers has been proposed. Simulation results show that the proposed algorithm can reduce the completion delay by over 30% with one-shot algorithm. We have also verified the robustness of the algorithm under different heterogeneous environments with disparate distributions of computation capabilities. In future work, we plan to take communication delay and power consumption into account and consider the efficiency of the multi-message communication strategy [15].

Appendix A
Proof of Theorem 1

The partial derivative of $\mathcal{L}\left(t_m^{[e]}, \alpha, \boldsymbol{t}_m^{[s]}\right)$ can be derived as:

$$\frac{\partial \mathcal{L}(t_m^{[e]}, \alpha, \boldsymbol{t}_m^{[s]})}{\partial l_{m,n}} = \frac{u_n(t_m^{[e]} - t_{m,n}^{[s]})}{l_{m,n}} e^{-\frac{u_n(t_m^{[e]} - t_{m,n}^{[s]})}{l_{m,n}} + a_n u_n} - 1 + e^{-\frac{u_n}{l_{m,n}}(t_m^{[e]} - t_{m,n}^{[s]} - a_n l_{m,n})}, \tag{14}$$

$$\frac{\partial \mathcal{L}(t_m^{[e]}, \alpha, \boldsymbol{t}_m^{[s]})}{\partial \alpha} = L_m - \sum_{n=1}^{N} l_{m,n}\left[1 - e^{-\frac{u_n}{l_{m,n}}\left(t_m^{[e]} - t_{m,n}^{[s]} - a_n l_{m,n}\right)}\right]. \tag{15}$$

We can get the relationship between $t_m^{[e]}$ and $l_{m,n}^*$ through (14) and get the equality: $l_{m,n}^* = \frac{t_m^{[e]} - t_{m,n}^{[s]}}{\phi_{m,n}}$.

Replacing $l_{m,n}$ in (15) with $t_m^{[e]}$:

$$L_m - \sum_{n=1}^{N} \frac{t_m^{[e]} - t_{m,n}^{[s]}}{\phi_{m,n}} \left(1 - \frac{1}{1 + u_n \phi_{m,n}}\right) = 0. \tag{16}$$

Finally, the expressions of $t_m^{[e]}$ and $l_{m,n}^*$ can be obtained.

References

[1] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: avoiding stragglers in distributed learning," in *Proc. Int. Conf. on Machine Learning*, Sydney, Australia, Aug. 2017, pp. 3368-3376.

[2] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "A unified coding framework for distributed computing with straggling servers," *IEEE Global Commun. Conf. Workshop*, Washington, DC, USA, Dec. 2016.

[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] E. Ozfatura, D. Gündüz, and S. Ulukus, "Speeding up distributed gradient descent by utilizing non-persistent stragglers," *Proc. IEEE Int. Symp. on Inform. Theory (ISIT)*, Paris, France, 2019, pp. 2729-2733.

[5] A. Reisizadeh, S. Prakash, R. Pedarsani and A. S. Avestimehr, "Tree Gradient Coding," *Proc. IEEE Int. Symp. on Inform. Theory (ISIT)*, Paris, France, 2019, pp. 2808-2812.

[6] N. Ferdinand and S. C. Draper, "Hierarchical Coded Computation," *Proc. IEEE Int. Symp. on Inform. Theory (ISIT)*, Vail, CO, 2018, pp. 1620-1624.

[7] B. Buyukates and S. Ulukus, "Timely Distributed Computation With Stragglers," *IEEE Trans. Commun.*, vol. 68, no. 9, pp. 5273-5282, Sept. 2020.

[8] R. Bitar, M. Wootters and S. El Rouayheb, "Stochastic Gradient Coding for Straggler Mitigation in Distributed Learning," *IEEE J. Sel. Areas Commun.*, vol. 1, no. 1, pp. 277-291, May 2020.

[9] M. Fahim and V. R. Cadambe, "Lagrange Coded Computing with Sparsity Constraints," *57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)* Monticello, IL, USA, 2019.

[10] M. Mohammodi Amiri, and D. Gündüz, "Computation scheduling for distributed machine learning with straggling workers," *IEEE Trans. Signal Process*,vol. 67, no. 24, pp. 6270-6284, Dec. 2019.

[11] Y. Sun, J. Zhao, S. Zhou and D. Gündüz, "Heterogeneous Coded Computation across Heterogeneous Workers," *IEEE Global Commun. Conf.*, Waikoloa, HI, USA, Dec. 2019.

[12] S. Li, S. M. M. Kalan, A. S. Avestimehr, and M. Soltanolkotabi, "Near-optimal straggler mitigation for distributed gradient methods," *IEEE Int. Parallel and Distributed Processing Symp*, Vancouver, BC, Canada, May 2018, pp. 857-866.

[13] A. Behrouzi-Far and E. Soljanin, "On the Effect of Task-to-Worker Assignment in Distributed Computing Systems with Stragglers," *56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)* Monticello, IL, USA, 2018.

[14] A. Reisizadeh, S. Prakash, R. Pedarsani and S. Avestimehr, "Coded computation over heterogeneous clusters," *Proc. IEEE Int. Symp. on Inform. Theory (ISIT)*, Aachen, 2017, pp. 2408-2412.

[15] E. Ozfatura, S. Ulukus, and D. Gündüz, "Straggler-aware Distributed Learning: Communication Computation Latency Trade-off," *Entropy*, 2020, 22(5), 544.