Coding for Heterogeneous UAV-based Networked Airborne Computing

Baoqian Wang, Junfei Xie, *Member, IEEE*, Kejie Lu, *Senior Member, IEEE*, Yan Wan, *Senior Member, IEEE,* and Shengli Fu, *Senior Member, IEEE*

***Abstract*—Considering the limited computing and storage capacities of a single unmanned aerial vehicle (UAV) with small payload, most existing small UAV-based applications execute computation-intensive tasks (e.g., image processing) at ground stations. However, offloading tasks to the ground is not suitable for delay-sensitive applications that require real-time responses. To address this challenge, this paper explores distributed computing techniques to enable networked airborne computing of high computing and storage capabilities. In particular, the matrix multiplication problem, one of the most basic building blocks of many computing tasks, is considered in this preliminary study. We then develop an innovative coding scheme for distributed computation over heterogeneous UAV clusters, which uses codes to substantially reduce the latency of computing and improve the robustness to system noises. It also adopts a batch processing procedure to relax the requirement of complete results enforced in existing solutions to further enhance the efficiency. Simulation and real experimental results show the promising performance of the proposed scheme compared with the uncoded scheme and the existing solution for coded computation over heterogeneous computing clusters.**

Keywords*—*Unmanned Aerial Vehicles, Airborne Computing, Distributed Computing, Coded Computation, Task Allocation

I. Introduction

The past few years have witnessed a rapid growth in the use of unmanned aerial vehicles (UAVs) to facilitate civilian and commercial applications including precision agriculture [1], emergency response [2], infrastructure monitoring [3], etc. In these applications, computation-intensive tasks such as 3-dimensional (3-D) mapping, precise positioning, path planning, etc., are currently conducted at ground stations, due to the limited computing and storage capability of a single UAV of small payload. However, such computation mechanism suffers from significant ground-air transmission delays or even failures and thus is unsuitable for delay-sensitive applications that require real-time responses. The recently proposed *UAV-based networked airborne computing* [4], [5] that allows computation-intensive tasks to be executed directly on-board of UAV through networking and resource sharing among multiple UAVs is envisioned to address these challenges, substantially improving system performance and enabling advanced new applications. However, it still has a long way to go.

In our previous efforts on developing the UAV-based networked airborne computing platform, we built a prototype that uses NVIDIA Jetson TX2 as the computing unit and adopts the directional antenna, Ubiquiti Nanostation LocoM2, to enable long-distance and broad-band UAV-to-UAV communications [6], [7]. This prototype implements virtualization for enhanced computing and resource management capabilities [5]. It also implements an advanced learning-based directional antenna control solution to achieve robust UAV-to-UAV communications [8]. In this paper, we aim to further extend the computing capability of this prototype by exploring distributed computing techniques.

Distributed computing has been widely studied in the literature since the last few decades [9], [10]. MapReduce [11] and Apache Spark [12] are the two prevalent modern distributed computing frameworks that allow processing of data in the order of petabytes. However, their performances are still not optimized, which are vulnerable to uncertain system noises, such as node failures, anomalous behaviors, communication bottlenecks, etc [13]. Note that this issue of system noise is more prominent in UAV networks with dynamic topology and high mobility.

The *coding* techniques, which have gained great success in improving the resilience of communication, storage and cache systems to uncertain system noises [14], recently attract increasing attention from the distributed computing community. Lee *et al.* [15], [16] first present the use of coding to speed up matrix multiplication and data shuffling, the two basic building blocks of distributed machine learning algorithms. Followed by this study, different computation problems have been explored using codes, such as the gradients [17], large matrix-matrix multiplication [18], linear inverse problems [19], nonlinear operations [20], etc. Other coded computation solutions include the "Short-Dot" coding scheme [21] that offers computation speed-up by introducing additional sparsity to the coded matrices and the unified coded framework [22], [23] that achieves the trade-off between communication load and computation latency. More relevant to our study, Reisizadeh *et al.* [24] investigate the coded matrix multiplication problem over heterogeneous computing clusters. The authors prove that the solution obtained by the proposed Heterogeneous Coded Matrix Multiplication (HCMM) algorithm is asymptotically optimal under the assumption that the processing time of each computing node follows an exponential or Weibull distribution.

Junfei Xie and Baoqian Wang are with the Department of Electrical and Computer Engineering, San Diego State University, San Diego, CA, 92182 (e-mail: jxie4@sdsu.edu)

Kejie Lu is with the Department of Computer Science and Engineering, University of Puerto Rico at Mayagüez, Mayagüez, Puerto Rico, 00681, email: (kejie.lu@upr.edu)

Yan Wan is with the Department of Electrical Engineering, University of Texas at Arlington, Arlington, Texas, 76019 (e-mail: yan.wan@uta.edu).

Shengli Fu is with the Department of Electrical Engineering, University of North Texas, Denton, Texas, 76201 (e-mail: Shengli.Fu@unt.edu).

This work was partially supported by National Science Foundation (NSF) under Grants CI-1730589/1730675/1730570/1730325 and CAREER-1714519.

978-1-7281-0960-2/19/$31.00 ©2019 IEEE

We note that a common characteristic of aforementioned coded computation solutions is that each computing node waits to send back the result until the whole task is completed, which may incur significant computation latency. Moreover, for many UAV applications that benefit from timely decisions such as emergency response, approximate solutions derived from partial results can be very useful. In this paper, we aim to tackle this challenge and develop a new coding scheme for distributed computation over heterogeneous UAV clusters, as a critical step towards realizing the networked airborne computing. This scheme is featured by a batch processing procedure that allows partial results to be returned early and also speeds up the computation. To illustrate the performance of the proposed scheme, we conduct both simulation studies and real experiments.

The rest of this paper is organized as follows. Section II briefly reviews the coded matrix multiplication technique. Section III provides the problem formulation and corresponding solution. The simulation and experimental results are presented in Section IV and Section V, respectively. Section VI concludes the paper.

II. Coded Matrix Multiplication

In this section, we briefly explain the mechanism of the coded computation technique in solving matrix multiplication problems and how it improves the robustness of distributed computing systems to uncertain system noises.

Consider a matrix multiplication problem $A_{r\times m}X_{m\times n}$, where $X_{m\times n} \in R^{m\times n}$ is the input matrix, $A_{r\times m} \in R^{r\times m}$ is pre-stored in the system, and $r, m, n \in \mathbb{Z}^+$ are positive integers representing number of rows or columns. Both matrices can be very large, making the calculation of $A_{r\times m}X_{m\times n}$ at a single computing node to be cost prohibitive.

To solve above problem, traditional distributed computing systems divide matrix $A_{r\times m}$ into a set of sub-matrices $A_{l_1\times m}, A_{l_2\times m}, \cdots, A_{l_N\times m}$, and pre-stores each sub-matrix $A_{l_i\times m} \in R^{l_i\times m}$ in a different computing node i, called *worker node*, where N is the total number of worker nodes and $\sum_{i=1}^{N} l_i = r$, $l_i \in \mathbb{Z}^+$. Upon receiving the input matrix $X_{m\times n}$, the *master node* sends matrix $X_{m\times n}$ to the worker nodes. Each worker node i then computes $B_{l_i\times n} = A_{l_i\times m}X_{m\times n}$ and returns the result to the master node. After all results are received, the master node aggregates the results and outputs $A_{r\times m}X_{m\times n} = [B_{l_1\times n}; B_{l_2\times n}; \cdots; B_{l_N\times n}]$. Note that the existence of system noises, such as malfunctioning nodes and communication bottlenecks, will defer or even fail the computation, as delay or loss of any $B_{l_i\times n}$, $i \in \{1, 2 \ldots, N\}$, will affect the calculation of the final result $A_{r\times m}X_{m\times n}$.

The coded computation technique alleviates the effect of system noises and thus speeds up the computation by introducing redundancy using the codes. In particular, it encodes the pre-stored matrix $A_{r\times m}$ to a larger matrix $\hat{A}_{q\times m}$ with more rows, i.e., $q > r$. If the Maximum Distance Separable (MDS) codes [20] are adopted, the encoded matrix $\hat{A}_{q\times m}$ can be computed by the following equation

$$\hat{A}_{q\times m} = H_{q\times r}A_{r\times m} \tag{1}$$

where $H_{q\times r}$ is the encoding matrix with any r rows being full-rank. Its entries can take any real values. Similar as the uncoded computation, the encoded matrix $\hat{A}_{q\times m}$ is then divided into N sub-matrices $\hat{A}_{l_1\times m}, \hat{A}_{l_2\times m}, \cdots, \hat{A}_{l_N\times m}$, where $\sum_{i=1}^{N} l_i = q$, and each worker node i calculates $\hat{B}_{l_i\times n} = \hat{A}_{l_i\times m}X_{m\times n}$.

Different from the uncoded computation, the master node does not need to wait for all worker nodes to complete their calculations, as it is able to recover $A_{r\times m}X_{m\times n}$ once the total number of rows of the received results equal to or larger than r. In particular, suppose $\mathcal{I}$ is the set of indices of the worker nodes that have returned the results $\hat{B}_{l_i\times n}$ to the master node, and $\sum_{i\in\mathcal{I}} l_i \geq r$. The master node can then recover $A_{r\times m}X_{m\times n}$ using any r rows of the received results $[\hat{B}_{l_i\times n}; \cdots; \hat{B}_{l_j\times n}]$, where $i, j \in \mathcal{I}$, using the following equation

$$A_{r\times m}X_{m\times n} = H_{r\times r}^{-1}\hat{B}_{r\times n} \tag{2}$$

where $\hat{B}_{r\times n}$ is a sub-matrix of $[\hat{B}_{l_i\times n}; \cdots; \hat{B}_{l_j\times n}]$ with r rows. $H_{r\times r}$ is the associated encoding matrix that satisfies

$$\hat{B}_{r\times n} = \hat{A}_{r\times m}X_{m\times n} = H_{r\times r}A_{r\times m}X_{m\times n} \tag{3}$$

III. Main Results

In the previous section, how to allocate the computation load to each worker node, i.e., how to determine the value of l_i, is not addressed. Directly assigning equal loads or randomly dividing the loads may not be optimal. In this section, we formulate the coded matrix multiplication problem over heterogeneous UAV clusters, and introduce a new scheme to optimally allocate computation loads.

A. Problem Formulation

Given a matrix $A_{r\times m}$, consider the scenario where a UAV (*master node*) wants to compute $A_{r\times m}X_{m\times n}$ for an input matrix $X_{m\times n}$. As the computing resources in this UAV is limited, it distributes the computation loads to its N neighboring UAVs (*worker nodes*) of different computing powers using the coded computation technique (with MDS codes). In particular, each worker node i computes $\hat{B}_{l_i\times n} = \hat{A}_{l_i\times m}X_{m\times n}$, where $\hat{A}_{l_i\times m}$ is a sub-matrix of the encoded matrix $\hat{A}_{q\times m}$ in (1) and $\sum_{i=1}^{N} l_i = q$, $l_i \in \mathbb{Z}^+$. We assume the UAVs are hovering in the air and have a fixed topology. The communication quality between two UAVs also does not change.

Unlike existing coded distributed implementations, where each worker node returns the result after the whole assigned task is completed, we here define a new implementation that allows *partial* results to be returned. In particular, we let each worker node i further equally divide $\hat{A}_{l_i\times m}$ row-wise into p_i sub-matrices of $\lceil \frac{l_i}{p_i} \rceil = b_i$ rows, named as *batches*, where $p_i \in \mathbb{Z}^+$, $p_i \leq l_i$. Upon receipt of the input matrix $X_{m\times n}$ from the master node, the worker node multiplies each batch with $X_{m\times n}$ and sends back the result immediately. Suppose the master node receives $s_i(t)$ batches from the worker node i by time t, where $s_i(t) \in \{1, 2, \ldots, p_i\}$, it can then recover the final result when $\sum_{i=1}^{N} s_i(t)b_i \geq r$.

Now let us formulate the problem to be solved in this study. Let $T_{k,i}$ be the processing time of each worker node i to compute k batches, $k \in \mathbb{Z}^+$. Here we assume $T_{k,i}$ follows a shifted exponential distribution similar as the one used in HCMM [24], and $T_{k,i}$ is independent from $T_{k',j}$, $\forall j \in \{1,2,\ldots,N\}$, $j \neq i$, $k' \in \mathbb{Z}^+$. Particularly, the cumulative distribution function (CDF) of $T_{k,i}$ is given by

$$Pr(T_{k,i} \leq t) = \begin{cases} 1 - e^{-\mu_i(\frac{t}{kb_i} - \alpha_i)} & \text{if } t \geq kb_i\alpha_i \\ 0 & \text{else} \end{cases} \tag{4}$$

where μ_i and α_i are straggling and shift parameters, respectively. We also assume that the data transmission time among the UAVs is negligible. Denote the waiting time for the master node to receive a decodable set of results as T, where $\sum_{i=1}^{N} s_i(T)b_i \geq r$. Our problem can then be formulated as finding the optimal values of $l_i \in \mathbb{Z}^+$, $\forall i \in \{1,2,\ldots,N\}$, such that the expected waiting time $E(T)$ is minimized. The mathematical formulation is provided as follows

$$\begin{aligned} \mathcal{P}_{\text{main}}: \underset{l_i, \forall i \in \{1,2,\ldots,N\}}{\text{minimize}} \quad & E[T] \\ \text{subject to} \quad & l_i \in \mathbb{Z}^+, \forall i \in \ldots \{1,2,,N\} \end{aligned} \tag{5}$$

B. Solution

To efficiently solve the problem formulated above, which requires exhaustive search over all possible solutions and thus is an NP hard problem [25], we here provide an alternative problem formulation. In particular, let $S_i(t) = s_i(t)b_i$ be the number of rows of results received by the master node from worker node i by time t, and $S(t) = \sum_{i=1}^{N} S_i(t)$ be the total amount of results received by the master node by time t. Then, instead of minimizing the expected time $E[T]$, we minimize a feasible time t, such that, by time t, the expected amount of results received by the master node are sufficient to recover the final result. Mathematically, this alternative problem formulation for $\mathcal{P}_{\text{main}}$ is given as follows:

$$\begin{aligned} \mathcal{P}_{\text{alt}}: \underset{l_i, \forall i \in \{1,2,\ldots,N\}}{\text{minimize}} \quad & t \\ \text{subject to} \quad & E[S(t)] \geq r \\ & l_i \in \mathbb{Z}^+, \forall i \in \{1,2,\ldots,N\} \end{aligned} \tag{6}$$

To solve $\mathcal{P}_{\text{alt}}$, we first calculate $E[S(t)]$. Define $F_{k,i}(t) = Pr(T_{k,i} \leq t)$. The time to receive k and $k+1$ batches at node i is $T_{k,i}$ and $T_{k+1,i}$, respectively. Therefore, the probability for the master node to receive exactly k batches from worker node i by time t is

$$\begin{aligned} Pr(T_{k,i} \leq t < T_{k+1,i}) =& Pr(t < T_{k+1,i}) - Pr(t < T_{k,i}) \\ =& Pr(T_{k+1,i} > t) - Pr(T_{k,i} > t) \\ =& (1 - Pr(T_{k+1,i} \leq t)) \\ & - (1 - Pr(T_{k,i} \leq t)) \\ =& F_{k,i}(t) - F_{k+1,i}(t) \end{aligned} \tag{7}$$

where $k \in \{1,2,\ldots,p_i\}$. Note that $F_{p_i+1,i}(t) = 0$. We can then calculate $E[S_i(t)]$ as follows:

$$\begin{aligned} E[S_i(t)] &= \sum_{k=1}^{p_i} kb_i \left(F_{k,i}(t) - F_{k+1,i}(t)\right) \\ &= \sum_{k=1}^{p_i} kb_i F_{k,i}(t) - \sum_{k=1}^{p_i} kb_i F_{k+1,i}(t) \\ &= \sum_{k=1}^{p_i} b_i F_{k,i}(t) \end{aligned} \tag{8}$$

$E[S(t)]$ can then be computed by:

$$\begin{aligned} E[S(t)] &= \sum_{i=1}^{N} E[S_i(t)] \\ &= \sum_{i=1}^{N} \sum_{k=1}^{p_i} b_i F_{k,i}(t) \\ &= \sum_{i=1}^{N} \sum_{k=1}^{p_i} b_i \left(1 - e^{-\mu_i(\frac{t}{kb_i} - \alpha_i)}\right) \\ &= \sum_{i=1}^{N} \left(l_i - b_i \sum_{k=1}^{p_i} e^{-\mu_i(\frac{t}{kb_i} - \alpha_i)}\right) \end{aligned} \tag{9}$$

With $E[S(t)]$, we then use the interior point method [26] to get a near optimal solution to $\mathcal{P}_{\text{alt}}$, which is a nonlinear optimization problem. The derived optimal load allocations $l_1^*, l_2^*, ..., l_N^*$ are real numbers and are thus rounded to integers. As l_i is typically large in real applications, the impact of this approximation can be ignored [25].

IV. Simulation Studies

In this section, we conduct a series of simulation studies to investigate the performance of the proposed coding scheme.

We implement our method using Matlab. The computing model (4) is used to simulate the processing of batches at the worker nodes. Particularly, we generate random numbers according to (4) as the processing times of the batches. For comparison, we also implement the HCMM [24] and the uncoded scheme that divides computation loads equally. Note that in the special case when $p_i = 1, \forall i \in \{1,2,...,N\}$, our approach is same as the HCMM. To evaluate their performances, we consider the following four computation scenarios:

1) **Scenario 1:** This scenario considers the problem of computing $r = 10^4$ inner products using $N = 3$ worker nodes. The straggling parameters are set to $\mu_1 = 10$, $\mu_2 = 20$ and $\mu_3 = 30$.
2) **Scenario 2:** This scenario considers a similar problem as Scenario 1, but having $r = 10^5$.
3) **Scenario 3:** This scenario considers the problem of computing $r = 10^4$ inner products using $N = 100$ worker nodes. The straggling parameters μ_i are randomly chosen from $\{1,2,...,100\}$.
4) **Scenario 4:** This scenario considers a similar problem as Scenario 3, but having $r = 10^5$.

In all scenarios, the task assigned to each worker node i is decomposed into $p_i = 5$ batches, and the shift parameter is set to $\alpha_i = \frac{1}{\mu_i}, \forall i \in \{1,2,...,N\}$.

As shown in Table I, solving the optimization problem $\mathcal{P}_{\text{alt}}$ in (6) is computationally efficient. Figure 1 shows the mean execution time of each scheme to obtain the final result, where the mean times are found by averaging over 100 simulations. As we can see from the figure, our method outperforms both benchmark schemes in all scenarios. Moreover, our method achieves performance improvement of up to 92% over the uncoded scheme and up to 37% over HCMM.

Table I. TIME TO SOLVE $\mathcal{P}_{\text{alt}}$ IN DIFFERENT SCENARIOS

Scenarios	Time (s)
Scenario 1	0.432
Scenario 2	0.443
Scenario 3	1.40
Scenario 4	2.4

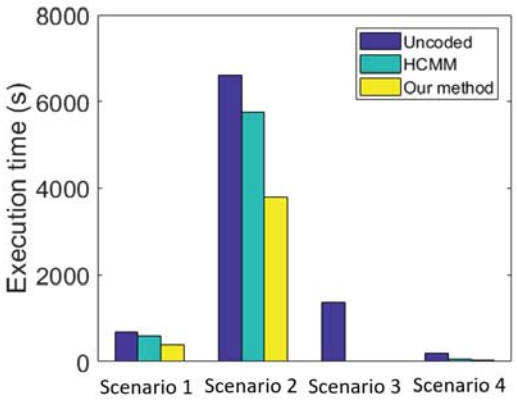

Figure 1. Comparison of the mean execution time of different load allocation schemes in the four scenarios.

To better demonstrate the advantages of our method, we also selectively show in Figure 2 the average total amount of inner product results received by the master node over time in Scenario 2 and Scenario 4. Remarkably, we can observe from Figure 2(a) that the master node in our method consistently receives results from the worker nodes from the very beginning. However, in both benchmark schemes, there is a large time slot at the beginning when the master node does not receive any results. This is because our method allows partial results to be returned, which is very useful for applications that require timely but unnecessarily optimized decisions. In Scenario 4 shown in Figure 2(b), the advantage of our method decreases, as more worker nodes are involved to share the computation loads.

In our method, the number of batches p_i is the parameter to be configured. To understand the impact of the parameter, we adopt the settings in Scenario 1 and vary the value of p_i, $\forall i \in \{1, 2, ...N\}$, which is configured to be same for all worker nodes. As shown in Figure 3, a larger p_i generally leads to better computation performance. However, selecting a larger p_i will introduce additional overheads such as the time to segment tasks into batches and to launch each batch processing program. We will leave the optimization of this parameter to the future work.

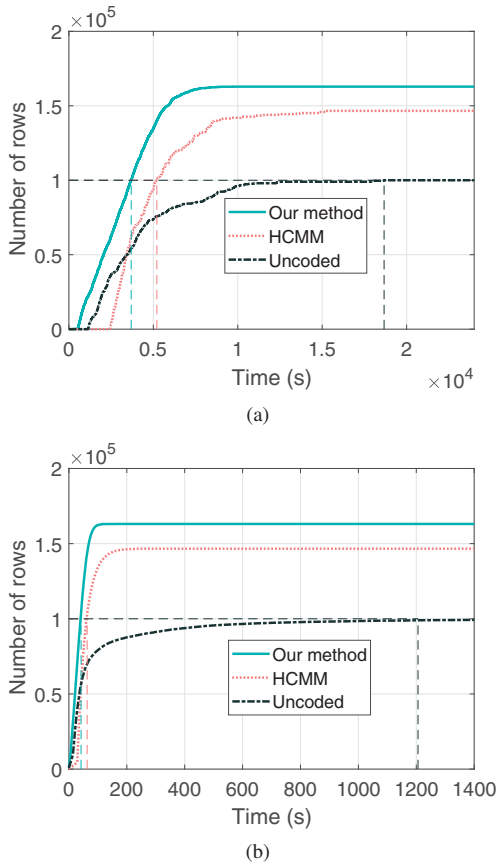

Figure 2. The average total number of rows of inner product results received by the master node over time in a) Scenario 2 and b) Scenario 4.

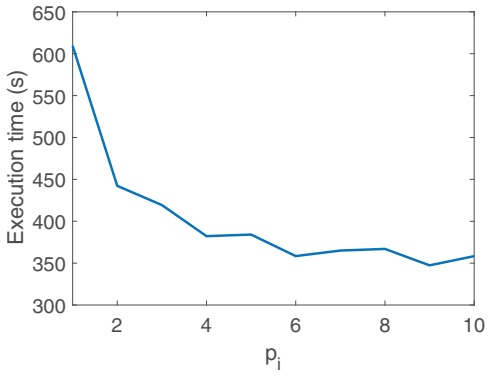

Figure 3. The mean execution time of the proposed coding scheme at different values of p_i in Scenario 1.

V. REAL EXPERIMENTAL STUDIES

In this section, we implement the proposed coding scheme on our prototype of the UAV-based networked airborne computing platform [5], [6], which adopts Jetson TX2 as the computing unit, implements Docker to virtualize computing resources, and uses LocoM2 to build UAV-to-UAV communications. We then conduct comparative real experiments to

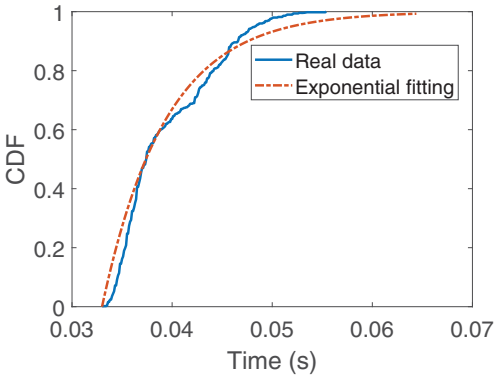

Figure 4. CDF of the execution time fitted with the shifted exponential distribution at $l = 2000$ and $c = 0.9$.

evaluate and demonstrate the performance of the proposed scheme.

A. Parameter Estimation

Before we conduct comparative studies, we first estimate the parameters of the processing time model (4) through model fitting. To measure the processing time required for a worker node of different computing powers to perform a matrix multiplication task of different sizes, we built a small distributed computing system of two nodes using the `mpi4py` package [27]. Specifically, we create two Docker containers on two different Jetson TX2, with one being the master node and the other being the worker node. The two nodes communicate through LocoM2 with bandwidth of $70Mbps$. We then let the master node send a matrix $X_{1000\times 2}$ to the worker node, which then multiplies $X_{1000\times 2}$ with a pre-stored matrix $A_{l\times 1000}$ and sends back the result. The roundtrip time is recorded for different values of l and different computing powers of the worker node, measured by the ratio c of the total resources of a single CPU core. Each experiment is repeated for 100 times.

Figure 4 shows an example cumulative distribution function (CDF) of the processing time at $l = 2000$ and $c = 0.9$, which fits the shifted exponential distribution well. The straggling parameter μ_i and the shift parameter α_i in the processing time model (4) are found to have the following relationships with the computing power c according to the results shown in Figure 5.

$$\mu_i = 3.03 \times 10^5 c^4 \tag{10}$$

$$\alpha_i = \frac{1.1861 \times 10^{-5}}{c} \tag{11}$$

B. Comparative Experimental Results

With the processing time model configured, we then implement the proposed coding scheme, as well as the uncoded scheme and HCMM for comparison. To evaluate their performances, we create a Docker container at one Jetson TX2 as the master node, and three containers at another Jetson TX2 as the worker nodes. Each worker node is assigned with $c = 0.1$, $c = 0.5$ and $c = 0.9$ of the total resources of a single CPU core, respectively. The corresponding straggling and shift parameters can then be calculated using (10)-(11). Under this setting, we then conduct an experiment with task size set to $r = 10^4$. In the experiment, the number of batches is set to $p_i = 5$, $\forall i \in \{1, 2, 3\}$. The experiment is repeated for 100 times and the average total number of rows of inner product results received by the master node are recorded. The comparison results shown in Figure 6 demonstrate the promising performance of our method, which match with the simulation results.

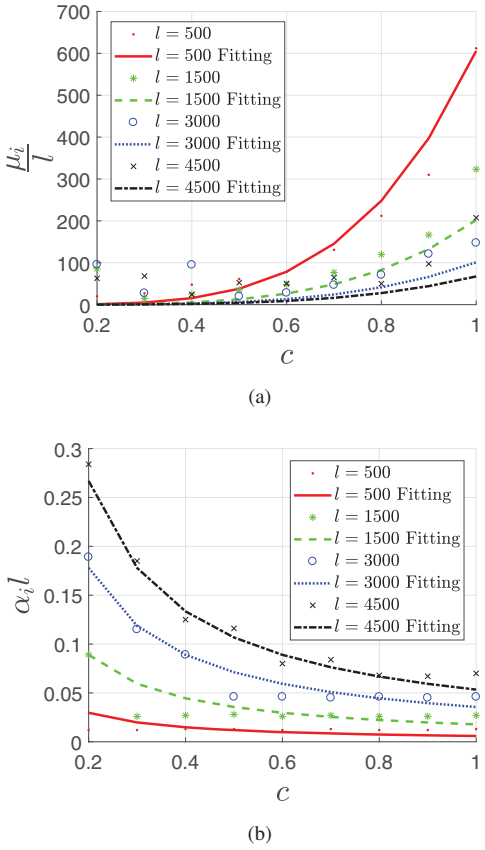

Figure 5. Models of the (a) straggling parameter μ_i and (b) shift parameter α_i fitted with experimental data obtained at different values of l and c.

VI. Conclusion

In this paper, we developed a new coding scheme for distributed computation over heterogeneous UAV clusters, as a step towards enabling UAV-based networked airborne computing. This scheme is featured by a batch processing procedure that allows partial results to be returned, which also significantly speeds up the computation. The simulation and real experimental results show that our method significantly outperforms the uncoded scheme and HCMM, the state-of-the-art coded computation solution over heterogeneous computing clusters. In the future, we will generalize our method to also consider energy consumption, transmission delays, and overhead caused by introducing batches.

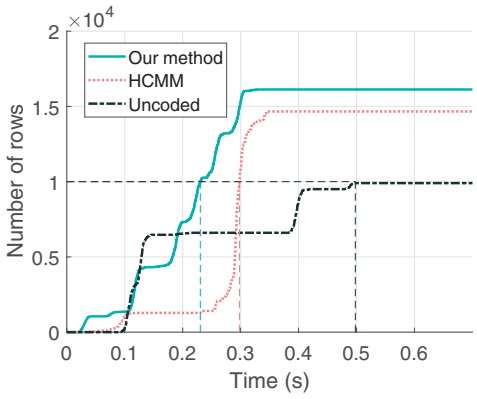

Figure 6. Comparison results of the real experiment with $r = 10^4$.

REFERENCES

[1] E. Honkavaara, H. Saari, J. Kaivosoja, I. Pölönen, T. Hakala, P. Litkey, J. Mäkynen, and L. Pesonen, "Processing and assessment of spectrometric, stereoscopic imagery collected using a lightweight uav spectral camera for precision agriculture," *Remote Sensing*, vol. 5, no. 10, pp. 5006–5039, 2013.

[2] K. Choi, I. Lee, J. Hong, T. Oh, and S. W. Shin, "Developing a uav-based rapid mapping system for emergency response," in *Proceedings of the 2009 Unmanned Systems Technology XI*. Orlando, Florida, USA: International Society for Optics and Photonics, April 2009.

[3] Y. Ham, K. K. Han, J. J. Lin, and M. Golparvar-Fard, "Visual monitoring of civil infrastructure systems via camera-equipped unmanned aerial vehicles (uavs): a review of related works," *Visualization in Engineering*, vol. 4, no. 1, p. 1, 2016.

[4] K. Lu, J. Xie, Y. Wan, and S. Fu, "Toward uav-based airborne computing," *IEEE Wireless Communications*, 2019.

[5] B. Wang, J. Xie, S. Li, Y. Wan, S. Fu, and K. Lu, "Enabling high-performance onboard computing with virtualization for unmanned aerial systems," in *Proceedings of the 2018 International Conference on Unmanned Aircraft Systems (ICUAS)*. Dallas, TX: IEEE, June 2018.

[6] "Airborne Computing Networks (Project Website)," Accessed: January 28, 2019. [Online]. Available: https://www.uta.edu/utari/research/robotics/airborne/

[7] J. Chen, J. Xie, Y. Gu, S. Li, S. Fu, Y. Wan, and K. Lu, "Long-Range and Broadband Aerial Communication Using Directional Antennas (ACDA): Design and Implementation," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 12, pp. 10 793–10 805, Dec 2017.

[8] S. Li, C. He, M. Liu, Y. Wan, Y. Gu, J. Xie, S. Fu, and K. Lu, "The Design and Implementation of Aerial Communication Using Directional Antennas: Learning Control in Unknown Communication Environment," *IET Control Theory & Applications*, 2019, accepted.

[9] S. Kartik and C. S. R. Murthy, "Task allocation algorithms for maximizing reliability of distributed computing systems," *IEEE Transactions on computers*, vol. 46, no. 6, pp. 719–724, 1997.

[10] B. Hong and V. K. Prasanna, "Distributed adaptive task allocation in heterogeneous computing environments to maximize throughput," in *Proceedings of the 18th International Parallel and Distributed Processing Symposium*. Santa Fe, NM, USA: IEEE, April 2004.

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

[12] 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.

[13] J. Dean and L. A. Barroso, "The tail at scale," *Communications of the ACM*, vol. 56, no. 2, pp. 74–80, 2013.

[14] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Coded mapreduce," in *Proceedings of the 2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. Monticello, IL, USA: UIUC, December 2015.

[15] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," in *Proceedings of the 2016 IEEE International Symposium on the Information Theory (ISIT)*. Barcelona, Spain: IEEE, July 2016.

[16] ——, "Speeding up distributed machine learning using codes," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1514–1529, 2018.

[17] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Proceedings of the 2017 International Conference on Machine Learning*. Sydney, Australia: IEEE, August 2017.

[18] K. Lee, C. Suh, and K. Ramchandran, "High-dimensional coded matrix multiplication," in *Proceedings of the 2017 IEEE International Symposium on Information Theory (ISIT)*. Aachen, Germany: IEEE, June 2017.

[19] Y. Yang, P. Grover, and S. Kar, "Coded distributed computing for inverse problems," in *Proceedings of the 2017 Advances in Neural Information Processing Systems*. CA, USA: IEEE, December 2017.

[20] K. Lee, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Coded computation for multicore setups," in *Proceedings of the 2017 IEEE International Symposium on Information Theory (ISIT)*. Aachen, Germany: IEEE, June 2017.

[21] S. Dutta, V. Cadambe, and P. Grover, "Short-dot: Computing large linear transforms distributedly using coded short dot products," in *Proceedings of the 2016 Advances In Neural Information Processing Systems*. Barcelona, Spain: IEEE, December 2016.

[22] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "A unified coding framework for distributed computing with straggling servers," in *Proceedings of the 2016 Globecom Workshops*. Washington, DC USA: IEEE, December 2016.

[23] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," *IEEE Transactions on Information Theory*, vol. 64, no. 1, pp. 109–128, 2018.

[24] A. Reisizadeh, S. Prakash, R. Pedarsani, and S. Avestimehr, "Coded computation over heterogeneous clusters," in *Proceedings of IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2017, pp. 2408–2412.

[25] A. Reisizadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr, "Coded computation over heterogeneous clusters," *IEEE Transactions on Information Theory*, vol. 65, no. 7, pp. 4227–4242, 2019.

[26] I. J. Lustig, R. E. Marsten, and D. F. Shanno, "Computational experience with a primal-dual interior point method for linear programming," *Linear Algebra and Its Applications*, vol. 152, pp. 191–222, 1991.

[27] W. D. Gropp, W. Gropp, E. Lusk, A. Skjellum, and A. D. F. E. E. Lusk, *Using MPI: portable parallel programming with the message-passing interface*. MIT press, 1999, vol. 1.