Diversity/Parallelism Trade-Off in Distributed Systems With Redundancy

Pei Peng, Emina Soljanin, *Fellow, IEEE*, and Philip Whiting

***Abstract*—Distributed computing enables *parallel* execution of smaller tasks that make up a large computing job. Its purpose is to reduce the job completion time. However, random fluctuations in task service times lead to straggling tasks with long execution times. Redundancy provides *diversity* that allows job completion when only a subset of redundant tasks is executed, thus removing the dependency on the straggling tasks. Under constrained resources (here, a fixed number of parallel servers), increasing redundancy reduces the available resources for parallelism. In this paper, we characterize the *diversity vs. parallelism* trade-off and identify the optimal strategy among replication, coding, and splitting, which minimizes the expected job completion time. We consider three common service time distributions and establish three models that describe the scaling of these distributions with the task size. We find that different distributions with different scaling models operate optimally at different redundancy levels, thus requiring very different code rates.**

***Index Terms*—Distributed systems, straggler mitigation, diversity and parallelism trade-off, erasure coding, service time scaling.**

I. Introduction

DISTRIBUTED parallel computing has become necessary for handling machine learning and other algorithms with ever increasing complexity and data requirements. This is because it provides simultaneous execution of smaller tasks that make up larger computing jobs. However, the large-scale sharing of computing resources causes random fluctuations in task service times [2]. Therefore, although executed in parallel, some tasks, known as stragglers, take much more time to complete, which consequently increases the job service time. Redundancy, in the form of simple task replication, and more recently, erasure coding, has emerged as a potentially powerful way to shorten the job execution time. Task redundancy allows job completion when only a subset of redundant tasks get executed, thus avoiding stragglers, see e.g. [3]–[18] and references therein. Redundancy provides diversity since job completion can be accomplished in different ways, e.g., when any fixed-size subset of tasks gets executed.

In distributed, parallel computing with redundancy, both parallelism and diversity are essential in reducing job service time. However, both parallelism and diversity are provided by the same limited system's resources dedicated to the job execution, e.g., a fixed number of servers. To understand this tension on the system's resources that parallelism and diversity bring about, let us consider two extreme ways to assign a job to n servers. One is *splitting* or maximum *parallelism* with no redundancy. Here, the job is divided equally among the n workers, and thus it gets completed when all workers execute their tasks. The other is n-fold *replication* or maximum *diversity*. Here, the entire job is given to each worker, and thus it gets completed when at least one of the workers executes its task. Roughly speaking, splitting (maximal parallelism) is appropriate for large jobs with almost deterministic service time (i.e., no straggling servers), and replication (maximal diversity) is appropriate for small jobs with highly variable service time (i.e., many straggling servers).

Given a fixed number of workers n, the general question is how much parallelism vs. diversity should be used. Consider a coding scheme where jobs are split in k tasks and encoded into $n \geq k$ s.t. execution of any k is sufficient for job completion. On the one hand, the smaller the k, the larger the task each server is given to execute. On the other hand, the smaller the k, the smaller the subset of tasks necessary for job completion. The choice of k thus dictates the trade-off between parallelism (increasing with k) and diversity (decreasing with k). We are here concerned with characterising the diversity vs. parallelism trade-off for different service time and task execution models, i.e., with finding an optimal k for a given n.

There is a large body of literature on replication and erasure codes for classical machine learning and other algorithms (see e.g. [11]–[15], [19]–[30] and references therein), and thus it is reasonable to assume that codes exist for many job types and any n and k combination. However, very little is known about what exact n and k should be selected in a given scenario in order to optimize a particular metric or goal of interest. When the goal is only to have the job completed by a certain time and it is known that at most ℓ workers will not respond by that time, then simply setting $k = n - \ell$ will achieve the goal. However, service time is a random variable, and one can only talk about the probability of task completion by a certain time [16], [31]–[33]. Therefore, the question one

Manuscript received October 5, 2020; revised May 9, 2021; accepted October 8, 2021. Date of publication November 15, 2021; date of current version January 20, 2022. This work was supported by the National Science Foundation under Grant CIF-1717314. An earlier version of this paper was presented in part at the 2020 IEEE International Symposium on Information Theory (ISIT) [1] [DOI: 10.1109/ISIT44484.2020.9174030]. *(Corresponding author: Pei Peng.)*

Pei Peng and Emina Soljanin are with the Department of Electrical and Computer Engineering, Rutgers, The State University of New Jersey, Piscataway, NJ 08854 USA (e-mail: pei.peng@rutgers.edu; emina.soljanin@rutgers.edu).

Philip Whiting is with the School of Engineering, Macquarie University, Sydney, NSW 2109, Australia (e-mail: philip.whiting@mq.edu.au).

Communicated by M. A. Maddah-Ali, Associate Editor for Communications.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIT.2021.3127920.

Digital Object Identifier 10.1109/TIT.2021.3127920

should ask is which k minimizes the expected job completion time.

Several recent papers asked how much redundancy should be used in distributed systems. In particular, [17], [34], [35] are solely concerned with replication systems, and their results do not easily extend to erasure coding system. Coding systems were considered in e.g., [3], [36], [37]. However, the system's resources were not assumed to be limited, and thus the diversity vs. parallelism trade-off was not addressed.

To study the diversity vs. parallelism trade-off in systems with limited resources, we need to know the service time probability density function (PDF) as well as how it *scales* (changes) with the size of the task. Various service time PDFs have been adopted in the literature. For theoretical analysis, Pareto distribution was used in e.g., [3], [38]–[43], Erlang was used in e.g., [44]–[49], Shifted-Exponential was used in e.g., [19], [36], [50], [51], the exponential distribution was used in [4], [37], [52], and a certain type of Bi-Modal distribution was used in e.g., [53], Some general classes of distributions (log-concave/convex) were considered in [17], [54]. The experiments with Amazon EC2 servers reported in [55, Fig. 2] show that the service time can be modeled by a Bi-Modal distribution (e.g., the one we use in this paper) or a heavy tail distribution (e.g., Pareto). On the other hand, a recent system-level work justifies the exponential distribution by considering their experimental results running on AWS [18].

There is no consensus on how the service time PDFs scale with the task size. For example, if the service time for some unit size task is Exponential, then some models assume that the service time for an s times larger task is also Exponential with the s times larger mean (i.e., scaled exponential [14], [19], [37]), while other models assume that it is Erlang (sum of s exponential PDFs). If the service time for some unit size task is Shifted-Exponential, then some models assume that the service time for s times larger task is also Shifted-Exponential, only with an s times larger shift [20], [36], [56]. More sophisticated models studied how job size changes the tail of the Pareto service time [3]. In this paper, we consider a number of common service time and scaling models. We find that different models operate optimally at very different levels of redundancy, and thus may require very different code rates. The contributions of the paper are stated in more detail in Sec. III, after the computing system model is given in detail.

The paper is organized as follows: In Sec. II, we present the system architecture and the models for the service time and its scaling. In Sec. III, we state the problem and summarize the contributions of this paper. In Sec. IV, V, and VI, we characterize the diversity vs. parallelism trade-off for three common service time distributions with three different scaling models. Conclusions are given in Sec. VII.

II. System Model and Problem Formulation

A. System Architecture

We adopt a system model as shown in Fig. 1, consisting of a single front-end master server and n computing servers we refer to as workers. Such distributed, parallel computing system architectures where a single master node manages a computing cluster of nodes are commonly implemented in modern frameworks, e.g., Kubernetes [57] and Apache Mesos [58].

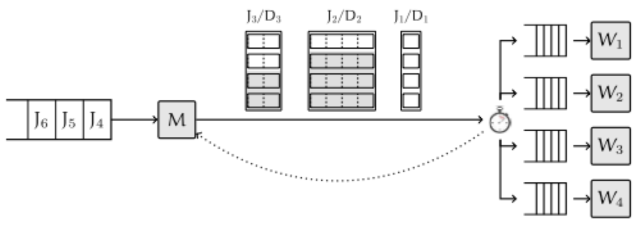

Fig. 1. A distributed computing system: Master node M partitions jobs J_i into tasks, possibly generates redundant tasks, and dispatches them to workers W_1, W_2, W_3, W_4. Shaded regions in the pre-processed jobs J_2, J_3 indicate redundancy. Here, each job consists of 4 computing units. Job J_1 is executed with maximal parallelism (splitting), J_2 with maximal diversity (replication), and J_3 is encoded by a $[4, 2]$ erasure code.

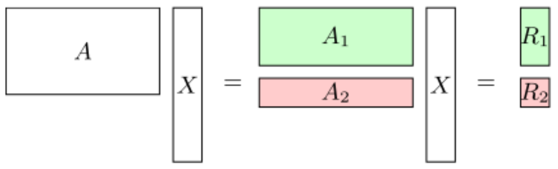

Fig. 2. Multiplication $A \cdot X$ of a 3-row matrix A by vector X is accomplished by parallel multiplication is of a 2-row submatrix A_1 by vector X and a 1-row submatrix A_2 by vector X.

B. Computing Jobs, Tasks, and Units

We are concerned with computing jobs that can be split into tasks which can then be executed independently in parallel by different workers. An example of such a job is vector by matrix multiplication, which is a basic operation in regression analysis and PageRank, and also in gradient descent, which resides at the core of almost any machine learning algorithm [59]–[66].

We assume that there is some minimum size task for a given job below which distributed computing would be inefficient, and refer to it as the *computing unit (CU)*. A task given to each worker can have one or more computing units. For example, if the job is to find a product $A \cdot X$ of a $3 \times n$ matrix A and an $n \times 1$ vector X, the computing unit could be a scalar product of a row of A and X. Let the matrix A be split into two submatrices: a $2 \times n$ submatrix A_1 and a $1 \times n$ submatrix A_2, as shown in Fig. 2. The job to compute $A \cdot X$, consisting of 3 CUs, can be split in two tasks for parallel execution: task $A_1 \cdot X$ with two CUs, and task $A_2 \cdot X$ with one CU. We will measure the sizes of jobs and tasks by the number of their computing units. Although CU corresponds to the execution of identical tasks on different data sets, their execution times are not necessarily identical nor identically distributed, as we discuss in more detail below.

C. Erasure Encoding Model

We assume that each job consists of n CUs where n is the number of workers. The master node partitions each job into k tasks, each of size $s = n/k$. It then generates $n - k$ redundant tasks, and dispatches the n tasks to the n workers. Therefore, each worker is assigned a task of $s = n/k$ CUs. The redundant tasks are generated by an erasure code. If an $[n, k]$ code with minimum distance d is used, then the job is completed when any $m = n - (d - 1)$ out of n tasks are completed. The Singleton bound imposes the constraint $m \geq k$, where $m = k$ for MDS codes. In this paper, we limit our analysis to MDS codes for two main reasons: 1) MDS are the most common codes used in the literature on performance analysis of erasure coded distributed systems and 2) MDS codes are sufficient to show the diversity/parallelism tradeoff and its dependence on multiple features in the system model, which is the purpose of this paper. However, our analysis is not limited to MDS codes, and we outline how it can be extended to general erasure codes in Section VII.

Fig. 1 shows some possible ways in which the master server can preprocess a job, i.e., partition a job into CUs, group the CUs into tasks, and add redundancy. Because of redundancy, not all tasks assigned to the workers will be executed or even partially serviced. Because of that, we refer to the preprocessed jobs as virtual demands. Consider the virtual demands D_1, D_2, and D_3 in the figure (resulting from processing jobs J_1, J_2, and J_3). Here, all jobs consist of 4 CUs. No redundant tasks are formed for job J_1, and thus the virtual demand D_1 and the original job are identical. Job J_2 is replicated on 4 workers, and thus the size of its virtual demand D_2 is 4 times the size of J_2. Job J_3 is encoded by a systematic $[4, 2]$ MDS code that generates 2 coded tasks of 2 CUs size. Its virtual demand D_3 is organized as follows: Workers W_1 and W_2 are each given a task consisting of 2 different CUs of J_3. Workers W_3 and W_4 are each given a coded task of 2 CUs size. Job J_1 is handled by splitting, J_2 by replication, and J_3 by coding. Roughly speaking, the goal of this paper is to determine which of these three strategies should be used for several service time models for executing single and multiple CUs used in the literature.

D. Computing Unit Service Time Models

We model a computing unit service time as a random variable (RV) X, and refer to the tasks that are still running after a given time as stragglers. As we discussed in the introduction, there is no consensus on what the probability distribution of X is. We adopt the following three service time models commonly used in the literature.

(Shifted)-Exponential: $X \sim \text{S-Exp}(\Delta, W)$: Support of X is $[\Delta, \infty)$, Δ is the minimum service time. The tail distribution is given as $\Pr\{X > x\} = e^{-(x-\Delta)/W}$ for $x > \Delta$. The larger the W the more likely straggling becomes. If $\Delta = 0$, then $X \sim \text{Exp}(W)$ is exponential.

Pareto: $X \sim \text{Pareto}(\lambda, \alpha)$: Support of X is $[\lambda, \infty)$ where λ is the minimum service completion time. Tail distribution is given as $\Pr\{X > x\} = (\frac{\lambda}{x})^\alpha$ for $x > \lambda$ where α is known as the tail index and models tail heaviness. Smaller α means a heavier tail, and thus more likely straggling.

Simple Bi-Modal: $X \sim \text{Bi-Modal}(B, \epsilon)$: Under this distribution, X takes only two values:

$$X = \begin{cases} 1 & \text{w.p. } 1 - \epsilon \\ B > 1 & \text{w.p. } \epsilon \leftarrow \text{probability of straggling} \end{cases} \tag{1}$$

This distribution features two important aspects of service straggling: probability of straggling ϵ and magnitude of straggling B.

E. Service Time Scaling With the Task Size

In the previous section, we listed three common models for the service time of a single computing unit. Together with those models, we will adopt three models for the service time of consecutive computing units that have frequently been used in the literature, as discussed in the introduction. The number of computing units that get assigned to each worker (that is, their task sizes) depends on the code rate k/n used in the system, and thus these scaling models are relevant because they tell us how the task service time scales with its size.

We consider three different commonly adopted models for the service time of consecutive CUs execution on the same server. For all three models, we assume independence across the servers. The models are described next, and their impact on the diversity vs. parallelism trade off is one of the main concerns of this paper.

Model 1 – *Server-Dependent Scaling:* The assumption here is that the straggling effect depends on the server and is identical for each CU executed on that server. Namely, there is some initial handshake time Δ after which the server completes its first and each subsequent CU in time X, i.e. $Y = \Delta + s \cdot X$. E.g. $X \sim \text{Exp}(W)$, then $Y \sim \text{S-Exp}(\Delta, sW)$. Note that Δ may be equal to 0, giving $Y = s \cdot X$. For example, when $X \sim \text{Pareto}(\lambda, \alpha)$, then $Y \sim \text{Pareto}(s\lambda, \alpha)$.

Model 2 – *Data-Dependent Scaling:* The assumptions here are that 1) each CU in a task of s CUs takes Δ time units to complete and 2) there are some inherent additive system randomness at each server which does not depend on the task size s that determines the straggling effect X. Therefore, $Y = s \cdot \Delta + X$.

Model 3 – *Additive Scaling:* The assumption here is that the execution times of CUs are i.i.d. Therefore, $Y = \sum_{i=1}^{s} X_i$ where $X_1, X_2, \cdots, X_s$ are independent.

F. Job Completion Time

As discussed above, the task execution times of the workers are i.i.d. RVs. The PDF of the RV Y modeling task execution time depends on the assumed model for the execution of a single and multiple CUs. When an $[n, k]$ code is used, the job is complete when any k out of n workers complete their size s tasks ($s = n/k$). Thus, the job completion time is also an RV, which we denote by $Y_{k:n}$ since it represents the k-th order statistic of n RVs distributed as Y. Let $Y_1, Y_2, \ldots, Y_n$ be n samples of some RV Y. Then the k-th smallest is an RV, commonly denoted by $Y_{k:n}$, and known as the k-th order statistic of $Y_1, \ldots Y_n$.

n –	number of workers and also the job size in CUs
k –	number of workers that have to execute their tasks for job completion (*diversity/parallelism parameter*)
s –	number of CUs per task, $s = n/k$
$Y_{k:n}$ –	job completion time when each worker's task size is s

III. Problem Statement and Summary of the Contributions

Our goal is to characterise the expected job completion time $\mathbb{E}[Y_{k:n}]$ for the service time and scaling models defined above. We are in particular interested in finding which k (i.e., code rate k/n) minimizes $\mathbb{E}[Y_{k:n}]$. Recall that when $k = 1$, we have replication (maximum diversity, no parallelism), and when $k = n$, we have splitting (maximum parallelism, no diversity). When $1 < k < n$, we use MDS coding and have a diversity/parallelism trade-off determined by the value of k.

The following table summarizes our findings by indicating whether splitting, replication, or coding minimizes the average job completion time $\mathbb{E}[Y_{k:n}]$. Much more detail is given in the following sections.

We consider the service time PDF and scaling models that are most commonly adopted in the literature. However, some of our results (we believe) can be extended to general service time PDFs, as we indicate by the claims and conjectures stated throughout the paper. These observations are relevant to practitioners who may have limited knowledge about their systems' behaviour.

To derive our results, we have relied on the following classical probabilistic models and arguments, which to the best of our knowledge, have not been previously used in this context. We introduce a generalized birthday problem to analyze the splitting strategy for the (Shifted-)Exponential service time with additive scaling. We recognize the stochastic dominance of splitting over coding. We show that the law of large numbers (LLN) can be used as an effective tool in finding the optimal code rate for systems with Bi-Modal service times and large number of workers. Moreover, we demonstrate how an LLN based analysis can be used to establish that for additive scaling and any service time distribution with the 4-th moment, splitting is a better strategy than replication for a sufficiently large number of workers.

IV. (Shifted-)Exponential Service Time

Under the (Shifted-)Exponential model, the CU service time is given by $\Delta + X$, where $X \sim \text{Exp}(W)$. The expected job completion time $\mathbb{E}[Y_{k:n}]$ depends on the service time of $s = n/k$ CUs, which is determined by the service time scaling model. In the following three subsections, we determine $\mathbb{E}[Y_{k:n}]$ for our three scaling models. Some results in this section were published in [1].

A. Server-Dependent Scaling

Under the server-dependent scaling, the service time of a task consisting of s CUs is given by $Y = \Delta + s \cdot X$, which means $Y \sim \text{S-Exp}(\Delta, sW)$. Therefore, the job completion time is given by

$$Y_{k:n} = \Delta + s \cdot X_{k:n}, \quad \text{where} \quad X \sim \text{Exp}(W)$$

and, by using the expression for $\mathbb{E}[X_{k:n}]$ in (19), we have

$$\mathbb{E}[Y_{k:n}] = \Delta + sW(H_n - H_{n-k}), \quad \text{where} \quad s = \frac{n}{k} \tag{2}$$

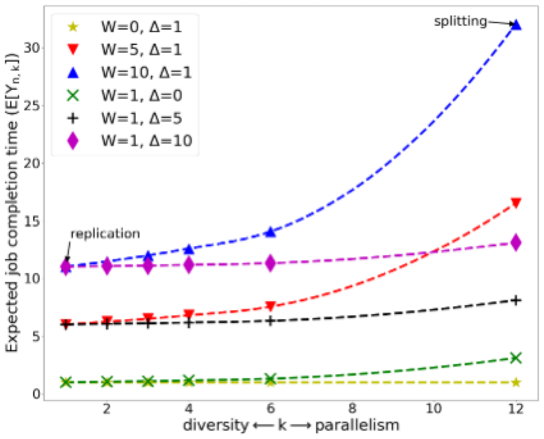

Fig. 3. Expected job completion time $\mathbb{E}[Y_{k:n}]$ for (Shifted-)Exponential service time with server-dependent scaling as a function of the diversity/parallelism parameter k (cf. (2)). The number of workers (job size) is $n = 12$, and task size per worker is $s = n/k$ (Since both k and s are integers, we have $k \in \{1, 2, 3, 4, 6, 12\}$. We use dashed curves to connect the points corresponding to different allowed values of k for a given Δ and W combination.) Replication is optimal for minimizing $\mathbb{E}[Y_{k:n}]$.

The minimum expected job completion time is given by the following theorem:

Theorem 1: The expected job completion time for $\text{S-Exp}(\Delta, W)$ service time with server-dependent scaling is minimized by replication (maximal diversity), i.e. $k = 1$.

Proof: From (2), we see that $\mathbb{E}[Y_{k:n}]$ is an increasing function of k for a given n, as follows

$$\begin{aligned}\mathbb{E}[Y_{k+1:n}] &= \Delta + W\frac{n}{k+1}(H_n - H_{n-k-1}) \\ &= \Delta + W\frac{n}{k+1}\Big(H_n - H_{n-k} + \frac{1}{n-k}\Big) \\ &= \mathbb{E}[Y_{k:n}] + \frac{Wn}{k+1}\Big[\frac{1}{n-k} - \frac{1}{k}(H_n - H_{n-k})\Big]\end{aligned}$$

Since the term in square brackets above is positive, we have $\mathbb{E}[Y_{k+1:n}] > \mathbb{E}[Y_{k:n}]$ for any positive integer $k \leq n$. □

Numerical Analysis: We evaluate (2) to see how the expected job completion time $\mathbb{E}[Y_{k:n}]$ changes with the diversity/parallelism parameter k. We consider a system with $n = 12$ workers and the following six different combinations of W and Δ: $\Delta = 1$ with $W \in \{0, 5, 10\}$, and $W = 1$ with $\Delta \in \{0, 5, 10\}$. The results are plotted in Fig. 3. $W = 0$ corresponds to the special scenario where $\mathbb{E}[Y_{k:n}] = \Delta$, that is, the service time is deterministic and does not change with k. When $W > 0$, $\mathbb{E}[Y_{k:n}]$ reaches its minimum at $k = 1$. When $W = 1$ and $\Delta \in \{0, 5, 10\}$, $\mathbb{E}[Y_{k:n}]$ increases with Δ, but changes little with k. When $\Delta = 1$ and $W \in \{0, 5, 10\}$, the slope of the corresponding curves increases with W. Although maximal diversity is optimal for all values of the parameters, it is much more effective in reducing the expected job completion time when W is large compared to when W is small.

B. Data-Dependent Scaling

Under the data-dependent scaling, the service time of a task consisting of s CUs is given by $Y = s \cdot \Delta + X$, which means

TABLE I

STRATEGIES THAT MINIMIZE THE AVERAGE JOB COMPLETION TIME FOR A GIVEN SERVICE TIME PDF AND SCALING MODEL

		SERVICE TIME PDF		
		Shifted Exponential	**Pareto**	**Bi-Modal**
SCALING	**Server-Dependent**	R [1]	S $\longrightarrow$ C [2]	S $\longrightarrow$ C $\longrightarrow$ S
	Data-Dependent	S $\longrightarrow$ C $\longrightarrow$ R	S $\longrightarrow$ C $\longrightarrow$ R	S $\longrightarrow$ C $\longrightarrow$ S
	Additive	S $\longrightarrow$ C	S $\longrightarrow$ C	S $\longrightarrow$ C $\longrightarrow$ S

[1] Strategies: R - replication, S - splitting, C - coding.
[2] $\longrightarrow$ indicates how the optimal strategy changes as the tail of the PDF becomes heavier (straggling becomes more likely).

$Y \sim$ S-Exp$(s\Delta, W)$, Therefore, the job completion time is given by

$$Y_{k:n} = s \cdot \Delta + X_{k:n}, \quad \text{where} \quad X \sim \text{Exp}(W)$$

and, by using the expression for $\mathbb{E}[X_{k:n}]$ in (19), we have

$$\mathbb{E}[Y_{k:n}] = s\Delta + W(H_n - H_{n-k}) = W\Big[\frac{n}{k}\cdot\frac{\Delta}{W} + (H_n - H_{n-k})\Big]. \tag{3}$$

Theorem 2: The expected job completion time for S-Exp(Δ, W) service time with data-dependent scaling is minimal when $k = k^*$, where $k^* = \arg\min_k W\left[\frac{nd}{k} + (H_n - H_{n-k})\right]$, $d = \Delta/W$. Furthermore, k^* takes the value $\lceil n(-d/2 + \sqrt{d + d^2/4})\rceil$ or $\lfloor n(-d/2 + \sqrt{d + d^2/4})\rfloor$.

Proof: The result is obtained by simple calculus using the log approximation to the harmonic numbers in (3). □

Note that this expression depends only on the ratio $d = \Delta/W$. For $\Delta \gg W$ (large d), the service time is essentially deterministic and it is optimal to use maximum parallelism, that is, splitting ($k = n$) is optimal. On the other hand, when $W \gg \Delta$ (small d) execution time is much more variable and it is optimal to operate with maximum diversity, that is, replication ($k = 1$) is optimal (cf. [36]).

Numerical Analysis: We evaluate (3) for $\mathbb{E}[Y_{k:n}]$ vs. k. We consider a system with $n = 12$ workers the following five different values of W/Δ: 1. $W = 0$ ($\Delta = 10$); 2. $W/\Delta = 0.1$ ($W = 1$, $\Delta = 10$); 3. $W/\Delta = 1$ ($W = 5$, $\Delta = 5$); 4. $W/\Delta = 10$ ($W = 10$, $\Delta = 1$); 5. $\Delta = 0$ ($W = 10$). The results are plotted in Fig. 4. By comparing different W/Δ scenarios, we conclude that when W/Δ (e.g. 0, 0.1) is small, then $\mathbb{E}[Y_{k:n}]$ decreases as k, increases, which means that splitting is optimal. When W/Δ is large (and $\Delta = 0$), the $\mathbb{E}[Y_{k:n}]$ increases with k, which means that replication is optimal. Otherwise, $\mathbb{E}[Y_{k:n}]$ reaches its minimum at $1 < k < 12$, which means that coding at a certain non-trivial rate is optimal. These observations are consistent with the theoretical analysis for k^*.

C. Additive Scaling

Under the additive scaling, the service time of a task consisting of s CUs is given by $Y = s\cdot\Delta + (X_1 + \cdots + X_s) = s\cdot\Delta + Z$, where $Z \sim \text{Erlang}(s, W)$. Therefore, the job completion time is given by

$$Y_{k:n} = s \cdot \Delta + Z_{k:n}, \quad \text{where} \quad Z \sim \text{Erlang}(s, W)$$

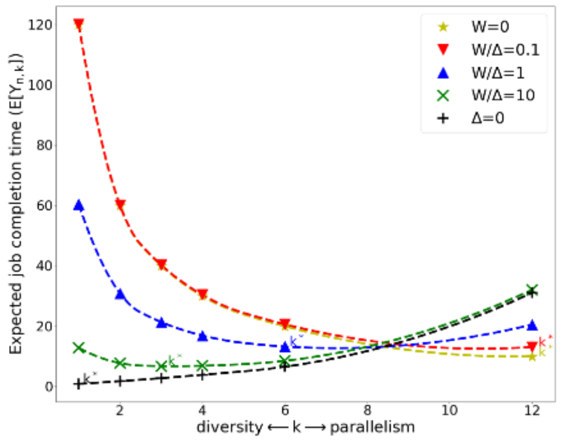

Fig. 4. Expected job completion time $\mathbb{E}[Y_{k:n}]$ for (Shifted-)Exponential service time with data-dependent scaling as a function of the diversity/parallelism parameter k (cf. (3)). The number of workers (job size) is $n = 12$, and task size per worker is $s = n/k$ (Since both k and s are integers, we have $k \in \{1, 2, 3, 4, 6, 12\}$. We use dashed curves to connect the points corresponding to different allowed values of k for a given Δ and W combination.) Parallelism outperforms diversity for small W/Δ and vice versa if W/Δ is large.

The expectation of the k-th order statistic of Erlang distribution is given by (20), which can be used for numerical results but is unsuitable for theoretical analysis. Asymptotics are available for large n and $k = \mathcal{O}(1)$. We now derive analytical expressions for the expected job completion time under splitting and replication, and show that splitting outperforms replication for sufficiently large n. We then show that rate $1/2$ coding outperforms splitting when $\Delta = 0$.

1) Splitting vs. Replication: Under splitting, the job completion time is given by

$$Y_{n:n} = \Delta + X_{1:n} + X_{1:(n-1)} + \cdots + X_{1:2} + X_{1:1} \tag{4}$$

where X_i's are i.i.d. Exp(W), then $X_{1:n}$ is Exp(W/n), and therefore,

$$\mathbb{E}[Y_{n:n}] = \Delta + WH_n$$

Under replication, we have

$$\mathbb{E}[Y_{1:n}] = n\Delta + W\frac{1}{n}\int_0^\infty e^{-t}\Big[R_n\Big(\frac{t}{n}\Big)\Big]^n dt$$

$$\text{where} \quad R_n(x) = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \cdots + \frac{x^{n-1}}{(n-1)!}$$

This result is a corollary of Theorem 3.

Theorem 3: Let the service times of CUs be independent and exponential with rate 1. If a job with d CUs is replicated over n workers, then the expected job completion time is

$$\frac{1}{n}\int_0^\infty e^{-t}\Big[S_d\Big(\frac{t}{n}\Big)\Big]^n dt \tag{5}$$

$$\text{where}\quad S_d(x) = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \cdots + \frac{x^{d-1}}{(d-1)!}$$

Proof: Let $t_1, t_2, \ldots$ be time epochs at which a CU gets completed on any of the n servers. Because all d CUs of the job are replicated on each of the n servers, the job is completed when d CUs get completed on any single server, which happens at some time $t_{\ell_{d,n}}$. Note that $\ell_{d,n}$ is a random variable. We represent $t_{\ell_{d,n}}$ as a sum of the CU inter-completion times.

$$t_{\ell_{d,n}} = \sum_{j=1}^{\ell_{d,n}}(t_j - t_{j-1}), \quad \text{where } t_0 \text{ is set to } 0. \tag{6}$$

Note that 1) $t_j - t_{j-1}$ are independent and exponentially distributed with rate n (the minimum of n independent exponentials with rate 1), and 2) $t_j - t_{j-1}$ are independent from $\ell_{d,n}$. Observe next that Wald's identity (Ch.10.2 in [67]) can be applied to (6). Therefore,

$$\mathbb{E}[t_{\ell_{d,n}}] = \frac{1}{n}\cdot\mathbb{E}[\ell_{d,n}]$$

Now observe that $\ell_{d,n}$ corresponds to a generalized birthday problem that the expected number of draws from n coupons until a coupon shows up d times. The claim follows from the result for $\mathbb{E}[\ell_{d,n}]$ in Appendix B. □

In Appendix B, we also have an asymptotic result for (5). For n large, we further simplify the expression of $\mathbb{E}[Y_{1:n}]$:

$$\mathbb{E}[Y_{1:n}] \sim n\Delta + \frac{W}{n}\sqrt[n]{n!}\,\Gamma(1+1/n)n^{1-\frac{1}{n}}, \quad \text{as} \quad n\to\infty \tag{7}$$

Theorem 4: For large enough n, splitting (maximal parallelism) outperforms replication (maximal diversity).

Proof: By using the Stirling's formula for $\sqrt[n]{n!}$ in (7), we have

$$\begin{aligned} & n\Delta + \frac{W}{n}\sqrt[n]{n!}\,\Gamma(1+1/n)n^{1-\frac{1}{n}} \\ & \geq n\Delta + \frac{W}{n}\sqrt[n]{\sqrt{2\pi}n^{n+\frac{1}{2}}e^{-n}}\,\Gamma(1+1/n)n^{1-\frac{1}{n}} \\ & = n\Delta + \frac{W}{n}\sqrt[n]{\sqrt{2\pi}n^{n+\frac{1}{2}}e^{-n}}\,\frac{\Gamma(2+1/n)}{1+1/n}n^{1-\frac{1}{n}} \\ & > n\Delta + \frac{W}{en}\sqrt[2n]{2\pi}\,n^{1+\frac{1}{2n}}n^{1-\frac{1}{n}}\frac{1}{1+1/n} > n\Delta + \frac{W}{2e}n^{1-\frac{1}{2n}} \end{aligned}$$

For a large enough n, $\mathbb{E}[Y_{1:n}]$ is well approximated by $n\Delta + \frac{W}{n}\sqrt[n]{n!}\,\Gamma(1+1/n)n^{1-\frac{1}{n}}$. Recall that $H_n = \mathcal{O}(\log n)$ and $n^{1-\frac{1}{2n}}/2e > \sqrt{n}/2e = \Omega(\sqrt{n})$. Therefore,

$$\mathbb{E}[Y_{1:n}] > \Delta + WH_n = \mathbb{E}[Y_{n:n}], \quad \text{as} \quad n\to\infty$$

Note that the theorem holds for $\Delta = 0$. □

2) Rate 1/2 Coding, $s = 2$: We consider the special case when $\Delta = 0$, n is even, and $s = n/k = 2$. Therefore, $k = n/2$ workers have to complete their two CUs in order for the job itself to be complete.

Let $Y_{n:n}$ be the time to complete the job under splitting, as given by (4) for $\Delta = 0$, and $Y_{n/2:n}$ the random time to complete the job under coding with $s = 2$. Theorem 5 below shows that $\mathsf{P}\{Y_{n/2:n} > x\} \leq \mathsf{P}\{Y_{n:n} > x\}$. It follows that

$$\mathbb{E}[Y_{n/2:n}] \leq \mathbb{E}[Y_{n:n}]$$

since for any non-negative random variable X, we have $\mathbb{E}[X] = \int_0^\infty \mathsf{P}(X > x)\,dx$.

It is, therefore, better to use a rate half code than splitting.

Theorem 5: Suppose that $n = 2k \geq 4$ is even. Then $Y_{n:n}$ stochastically dominates $Y_{n/2:n}$, that is,

$$\mathsf{P}\{Y_{n/2:n} > x\} \leq \mathsf{P}\{Y_{n:n} > x\}$$

Proof: Consider the system with $s = 2$ where scheduling until job completion is done as follows. The system runs until one server completes the first of its two CUs, at which point it is halted. This happens at a random time distributed as $X_{1:n}$. The system of the remaining $n-1$ servers runs until one server completes the first of its 2 CUs, at which point it is halted. This happens at a random time distributed as $X_{1:(n-1)}$ measured from the moment the first server was halted. The process continues in the same manner until $k = n/2$ servers have completed the first of their 2 CUs, at which point all remaining servers are halted. This happens at a random time T_1 given as

$$T_1 = X_{1:n} + X_{1:(n-1)} + \cdots + X_{1:(n-k+1)}$$

At this point, the $n - k = k$ servers which have completed one CU are restarted. The job is complete when each server completes the remaining CU, which happens at a random time T_2 given as

$$T_2 = X_{1:(n-k)} + X_{1:(n-k-1)} + \cdots + X_{1:1}$$

Note that, because some servers are halted, this system cannot perform better than the original $s = 2$ system. On the other hand, it performs as well as the $s = 1$ system since we have $Y_{n:n} = T_1 + T_2$. □

3) Numerical Analysis: We evaluate the derived expression for $\mathbb{E}[Y_{k:n}]$, and the results are shown in Fig. 5. We see that when W/Δ is small (e.g. 0, 0.1), splitting (maximum parallelism) gives the best performance. On the other hand, when W/Δ is large (e.g. 1, 10, ∞), we need coding in order to be optimal. The figure confirms Theorem 4 and Theorem 5 that say that splitting is better than replication and the rate half code is better than splitting when $\Delta = 0$.

Under the additive model, parallelism outperforms diversity, which was not always the case under the server-dependent and data-dependent models. Coding is optimal for some values of Δ and W, and the optimal code rate is around $1/2$.

V. Pareto Service Time

Under the Pareto model, the CU service time is given by X, where $X \sim \text{Pareto}(\lambda, \alpha)$. The expected job completion

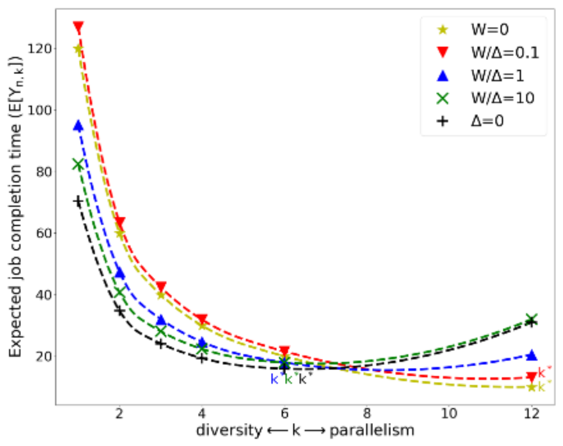

Fig. 5. Expected job completion time $\mathbb{E}[Y_{k:n}]$ for (Shifted-)Exponential service time with additive scaling as a function of the diversity/parallelism parameter k (cf. (20)). The number of workers (job size) is $n = 12$, and task size per worker is $s = n/k$ (Since both k and s are integers, we have $k \in \{1, 2, 3, 4, 6, 12\}$. We use dashed curves to connect the points corresponding to different allowed values of k for a given Δ and W combination.) When W/Δ is, splitting (maximal parallelism) is the best. When W/Δ is large, there is a balance between diversity and parallelism.

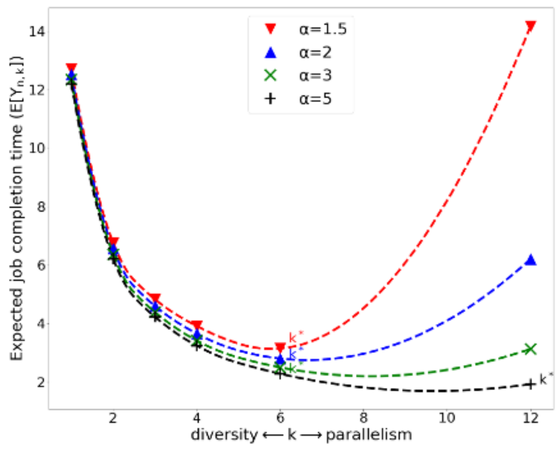

Fig. 6. Expected job completion time $\mathbb{E}[Y_{k:n}]$ for Pareto service time with server-dependent scaling as a function of the diversity/parallelism parameter k (cf. (8)). The number of workers (job size) is $n = 12$, and task size per worker is $s = n/k$ (Since both k and s are integers, we have $k \in \{1, 2, 3, 4, 6, 12\}$. We use dashed curves to connect the points corresponding to different allowed values of k for a given α.) The Pareto scale parameter is $\lambda = 1$. Splitting or rate 1/2 coding is optimal according to different α.

time $\mathbb{E}[Y_{k:n}]$ depends on the service time of $s = n/k$ CUs, which is determined by the service time scaling model. We next determine $\mathbb{E}[Y_{k:n}]$ for our three scaling models.

A. Server-Dependent Scaling

Under the server-dependent scaling, the service time of a task consisting of s CUs is given by $Y = s \cdot X$. Therefore, the job completion time is given by

$$Y_{k:n} = s \cdot X_{k:n}, \quad \text{where} \quad X \sim \text{Pareto}(\lambda, \alpha)$$

and, by using the expression for $\mathbb{E}[X_{k:n}]$ in (21), we have

$$\mathbb{E}[Y_{k:n}] = s\lambda \frac{n!}{(n-k)!} \frac{\Gamma(n-k+1-1/\alpha)}{\Gamma(n+1-1/\alpha)} \tag{8}$$

The minimum expected job completion time is given by the following theorem:

Theorem 6: The expected job completion time for $\text{Pareto}(\lambda, \alpha)$ service time with server-dependent scaling reaches the minimum when k^* is the ceiling or floor of $\frac{\alpha n-1}{\alpha+1}$.

Proof: From the definition of Gamma function, we have $\Gamma(z+1) = z\Gamma(z)$, and thus

$$\mathbb{E}[Y_{k:n}] = s\lambda \prod_{i=0}^{k-1} \frac{n-i}{n-1/\alpha-i} = \frac{n\lambda}{k} \prod_{i=0}^{k-1} \frac{n-i}{n-1/\alpha-i}$$

Therefore,

$$\frac{\mathbb{E}[Y_{k:n}]}{\mathbb{E}[Y_{k+1:n}]} = \frac{(k+1)(n-1/\alpha-k)}{k(n-k)}.$$

From this ratio, we see that when $k \leq \frac{\alpha n-1}{\alpha+1}$, then $\mathbb{E}[Y_{k:n}] \geq \mathbb{E}[Y_{k+1:n}]$, and when $k \geq \frac{\alpha n-1}{\alpha+1}$, then $\mathbb{E}[Y_{k:n}] \leq \mathbb{E}[Y_{k+1:n}]$. Since k is an integer, the minimum $\mathbb{E}[Y_{k:n}]$ is reached by setting k to the ceiling or floor of $\frac{\alpha n-1}{\alpha+1}$. □

Pareto distribution has a finite mean only when $\alpha > 1$. As α, the tail index, decreases, the right tail of Pareto becomes heavier. From Theorem 6, we know that the optimal k (k^*) increases with α. When $\alpha \downarrow 1$, $k^* \approx \lceil \frac{n-1}{2} \rceil$ or $\lfloor \frac{n-1}{2} \rfloor$, and $\mathbb{E}[Y_{k:n}]$ is minimized by coding. When $\alpha \to \infty$, then $k^* \approx n$, and $\mathbb{E}[Y_{k:n}]$ is minimized by splitting. Recall that $\alpha \to \infty$ implies an almost deterministic distribution, where splitting is expected to be optimal.

Numerical Analysis: We evaluate $\mathbb{E}[Y_{k:n}]$ to see how the expected job completion time changes with k. We consider a system with $n = 12$ workers for four different values of $\alpha \in \{1.5, 2, 3, 5\}$. We assume the Pareto scale parameter is $\lambda = 1$. The results are plotted in Fig. 6. When the tail is heavy ($\alpha = 1.5$), then $\mathbb{E}[Y_{k:n}]$ reaches its minimum at $k = 6$, and coding with the rate $1/2$ is optimal. Both replication and splitting have poor performance in this case. When the tail is light ($\alpha = 5$), then $\mathbb{E}[Y_{k:n}]$ is minimized by splitting. Replication still performs poorly. Otherwise ($\alpha = 2, 3$), coding with the rate $1/2$ is optimal. Splitting performs better than replication. From Theorem 6, we calculate the optimal $k^* = 6.8, 7.7, 8.8$, and 9.8 respectively for the four scenarios. Since $k \in \{1, 2, 3, 4, 6, 12\}$, k^* is either 6 or 12. The theoretically optimal k^*'s are consistent with the results in Fig. 6.

B. Data-Dependent Scaling

Under the data-dependent scaling, the service time of a task consisting of s CUs is given by $Y = s \cdot \Delta + X$. Therefore, the job completion time is given by

$$Y_{k:n} = s \cdot \Delta + X_{k:n}, \quad \text{where} \quad X \sim \text{Pareto}(\lambda, \alpha)$$

and, by using the expression for $\mathbb{E}[X_{k:n}]$ in (21), we have

$$\mathbb{E}[Y_{k:n}] = s\Delta + \lambda \frac{n!}{(n-k)!} \frac{\Gamma(n-k+1-1/\alpha)}{\Gamma(n+1-1/\alpha)} \tag{9}$$

We cannot easily derive the k^* that minimizes $\mathbb{E}[Y_{k:n}]$ from the above equation. However, according to the approximation

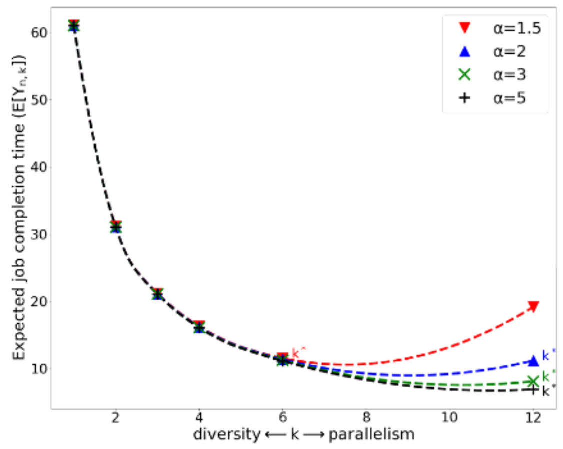

Fig. 7. Expected job completion time $\mathbb{E}[Y_{k:n}]$ for Pareto service time with data-dependent scaling as a function of the diversity/parallelism parameter k (cf. (9)). The number of workers (job size) is $n = 12$, the shift parameter is $\Delta = 5$ and task size per worker is $s = n/k$ (Since both k and s are integers, we have $k \in \{1, 2, 3, 4, 6, 12\}$. We use dashed curves to connect the points corresponding to different allowed values of k for a given α.) The Pareto scale parameter is $\lambda = 1$. Splitting or coding is optimal depending on the value of α.

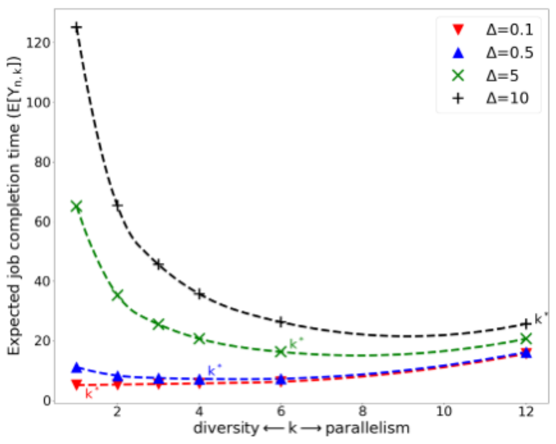

Fig. 8. Expected job completion time $\mathbb{E}[Y_{k:n}]$ for Pareto service time with data-dependent scaling as a function of the diversity/parallelism parameter k (cf. (9)). The number of workers (job size) is $n = 12$, and task size per worker is $s = n/k$ (Since both k and s are integers, we have $k \in \{1, 2, 3, 4, 6, 12\}$. We use dashed curves to connect the points corresponding to different allowed values of k for a given Δ.) The Pareto scale parameter is $\lambda = 5$ and shape parameter is $\alpha = 3$. The optimal code rate increases with Δ.

for the ratio of two gamma functions given in (22), we have

$$\mathbb{E}[Y_{k:n}] \approx \frac{n\Delta}{k} + \lambda\Big(\frac{n}{n-k}\Big)^{1/\alpha}.$$

Notice that the first term decreases with k, calling for maximal parallelism, whereas the second term increases with k, calling for maximal diversity.

Since the service time $Y = s \cdot \Delta + X$, we can understand Y as a shifted Pareto RV. Thus, Y can be approximated to a constant by Pareto RV based on whether the shift $s\Delta$ is far larger or smaller than the mean of Pareto $\frac{\alpha\lambda}{\alpha-1}$. As for the Shifted-Exponential distribution with data-dependent scaling, we conclude the following. When $\Delta \gg \frac{\alpha\lambda}{\alpha-1}$, i.e., $\mathbb{E}[Y_{k:n}] \approx \frac{n\Delta}{k}$, it is optimal to operate in the maximal parallelism mode. When $\Delta \ll \frac{\alpha\lambda}{\alpha-1}$, i.e., $\mathbb{E}[Y_{k:n}] \approx \lambda(\frac{n}{n-k})^{1/\alpha}$, it is optimal to operate in the maximal diversity mode.

Remark: We have concluded here for the Pareto service time and in the previous section for the exponential service time that replication is optimal when $\Delta \ll \mathbb{E}[X]$, splitting is optimal when $\Delta \gg \mathbb{E}[X]$, and coding is optimal otherwise. That conclusion applies to any service time PDF which has the first moment and the Δ and X components as in the cases considered here. The optimal code rate depends on how Δ compares to $\mathbb{E}[X]$.

Numerical Analysis: We evaluate $\mathbb{E}[Y_{k:n}]$ to see how the expected job completion time changes with the values of k. We consider a system with $n = 12$ workers for four different values of $\alpha \in \{1.5, 2, 3, 5\}$. The results are plotted in Fig. 7. We conclude that splitting is optimal when the Pareto tail is light (α is large) and coding becomes optimal when the tail gets heavier (α is small).

In Fig. 8, we consider the same system for four different values of $\Delta \in \{0.1, 0.5, 5, 10\}$. It is easy to calculate the Pareto mean $\frac{\alpha\lambda}{\alpha-1} = 7.5$. When $\Delta \ll 7.5$ (e.g. 0.1, 0.5), replication or low-rate coding is optimal. When Δ approaches 7.5 (e.g. 5, 10), splitting or high-rate coding is optimal. These observations validate the above analysis that the optimal strategy changes with the ratio of Δ and the Pareto mean.

C. Additive Scaling

Under the additive scaling, the service time of a task consisting of s CUs is given by

$$Y = X_1 + \cdots + X_s \quad \text{where} \quad X_i \sim \text{Pareto}(\lambda, \alpha)$$

and the job completion time is $Y_{k:n}$.

For splitting ($s = 1$), Y is a Pareto RV. Thus, the job completion time $\mathbb{E}[Y_{n:n}] = \mathbb{E}[X_{n:n}] = \lambda n! \frac{\Gamma(1-1/\alpha)}{\Gamma(n+1-1/\alpha)}$ [68]. For coding and replication, the corresponding expressions are difficult to derive. Nevertheless, we can compare splitting and replication when n is large by using the Law of Large Numbers (LLN). We conclude in Theorem 7 that splitting outperforms replication when n is sufficiently large.

Theorem 7: Under the Pareto(λ, α) service time with additive scaling, if the 4-th moment exists, i.e., if $\alpha > 4$, when n is sufficiently large, we have $\mathbb{E}[Y_{1:n}] > \mathbb{E}[Y_{n:n}]$, which means splitting outperforms replication by achieving a lower expected job completion time.

Proof: We prove that $\mathbb{E}[Y_{1:n}] > \mathbb{E}[Y_{n:n}]$ for $n \to \infty$, by showing that there is a function $f(n)$ such that

$$\mathbb{E}[Y_{n:n}] \le f(n) < \mathbb{E}[Y_{1:n}] \tag{10}$$

We first show that there is a function $f(n)$ which satisfies $\mathbb{E}[Y_{1:n}] > f(n)$ when $n \to \infty$. For the Pareto(λ, α) distribution, the mean $m_{\lambda,\alpha} = \lambda\alpha/(\alpha - 1)$ exist if $\alpha > 1$. Therefore, by the LLN, we have $\frac{1}{n}\sum_{j=1}^{n} X_j \to m_{\lambda,\alpha}$ *as* $n \to \infty$.

If $\alpha > 4$, the 4-th moment of Pareto distribution exists. Let $\mathbb{E}[X^4] = \xi < \infty$ and $Z = X - m_{\lambda,\alpha}$. By applying Jensen's inequality and removing negative terms, it follows that $\mathbb{E}[Z^4] = \mathbb{E}[X^4 - 4X^3 m_{\lambda,\alpha} + 6X^2 m_{\lambda,\alpha}^2 - 4m_{\lambda,\alpha}^3 X] + m_{\lambda,\alpha}^4 = \mathbb{E}[X^4 + 6X^2 m_{\lambda,\alpha}^2] - 4\,\mathbb{E}[X^3] m_{\lambda,\alpha} - 3m_{\lambda,\alpha}^4 \le 7\xi$. Next, define an RV S to be $S = \sum_{i=1}^{n} Z_i$. Since Z_i are i.i.d. (for $i = 1, 2, \ldots, n$) and $\mathbb{E}[Z_i] = 0$, by expanding S^4, we know the terms which have 4 or 3 different indices have 0 expectation. For example, $\mathbb{E}[Z_1^2 Z_2 Z_3] = \mathbb{E}[Z_1^2]\,\mathbb{E}[Z_2]\,\mathbb{E}[Z_3] = 0$. The terms which have 2 different indices have the expectation $\mathbb{E}[Z_i^2 Z_j^2] \le \xi$ by Jensen's inequality. And the coefficient is $3n^2 - 3n$. Similarly, the terms which have 1 index have the expectation $\mathbb{E}[Z_i^4] \le \xi$ and the coefficient n. Thus we have $\mathbb{E}[S^4] \le 3 \cdot 7\xi \cdot n^2$. Furthermore, by Markov's inequality, we obtain

$$p_n = \Pr\{|S|/n \ge \eta\} \le \frac{\mathbb{E}[S^4]}{n^4 \eta^4} \le \frac{21\xi}{n^2 \eta^4}, \tag{11}$$

where $|S|$ is the absolute value of S, and η is a small positive number. Note that $S + nm_{\lambda,\alpha} = \sum_{i=1}^{n} X_i$ is the time a worker takes to complete his task under the replication strategy. Let $S_i + nm_{\lambda,\alpha}$ be the time that the i-th worker takes to complete his task. Then, the job completion time is $Y_{1:n} = nm_{\lambda,\alpha} + \min_i S_i$. Since $S_1, \ldots, S_n$ are independent, we have that $\Pr\{Y_{1:n} > n(m_{\lambda,\alpha} - \eta)\} = \Pr\{Y_{1:n} - nm_{\lambda,\alpha} > -n\eta)\} = (\Pr\{S_1 > -n\eta\})^n > (1 - \Pr\{|S|/n \ge \eta\})^n$.

By using the bound in (11), it follows that

$$\Pr\{Y_{1:n} > n(m_{\lambda,\alpha} - \eta)\} > (1 - p_n)^n \ge \left(1 - \frac{21\xi}{n^2\eta^4}\right)^n \tag{12}$$

When $n \to \infty$, the rightmost term above tends to 1. Therefore, when n is sufficiently large, we have that $\Pr\{Y_{1:n} > n(m_{\lambda,\alpha} - \eta)\}) \to 1$, and thus $\mathbb{E}[Y_{1:n}] > n(m_{\lambda,\alpha} - \eta)$. Let $f(n) = n(m_{\lambda,\alpha} - \eta)$.

We next prove that $f(n) = n(m_{\lambda,\alpha} - \eta)$ satisfies $f(n) > \mathbb{E}[Y_{n:n}]$ for $n \to \infty$. The job completion time for splitting is $Y_{n:n} = \max_{1 \le i \le n} X_i$. Since $\alpha > 4$, the second moment of Pareto distribution exists. Let $\mathbb{E}[X^2] = \zeta < \infty$. Then, by Markov's inequality, we have

$$q_n = \Pr\{X \ge n(m_{\lambda,\alpha} - \eta)\} \le \frac{\zeta}{n^2(m_{\lambda,\alpha} - \eta)^2} \tag{13}$$

Since $X_1, \ldots, X_n$ are independent, we have that $\Pr\{Y_{n:n} < n(m_{\lambda,\alpha} - \eta)\} = (\Pr\{X_1 < n(m_{\lambda,\alpha} - \eta)\})^n = (1 - \Pr\{X \ge n(m_{\lambda,\alpha} - \eta)\})^n$. By using the bound in (13), it follows that $\Pr\{Y_{n:n} < n(m_{\lambda,\alpha} - \eta)\} = (1 - q_n)^n \ge \left(1 - \frac{\zeta}{n^2(m_{\lambda,\alpha} - \eta)^2}\right)^n$

When $n \to \infty$, the rightmost term above tends to 1. Therefore, when n is sufficiently large, we have $\Pr\{Y_{n:n} < n(m_{\lambda,\alpha} - \eta)\} \to 1$, and thus $\mathbb{E}[Y_{n:n}] < n(m_{\lambda,\alpha} - \eta) = f(n)$.

We have shown that when n is sufficiently large, the inequalities (10) hold, which proves the theorem. □

From the proof of Theorem 7, we get the lower bound on the expected time for replication: $\mathbb{E}[Y_{1:n}] \ge n(m_{\lambda,\alpha} - \eta) r_n$. Here η is a small and positive and $r_n = (1 - \frac{21\xi}{n^2\eta^4})^n$ by (12).

Observe that having Pareto service time plays no special role, and the arguments we used apply to any PDF which has the 4-th moment, as we formally express in Corollary 1.

Corollary 1: For a general service time distribution with the fourth moment, splitting results in a smaller expected job completion time than replication under additive scaling when the number of workers is sufficiently large.

Simulation Analysis: Although the expression of $\mathbb{E}[Y_{k:n}]$ is unknown, we can analyze $\mathbb{E}[Y_{k:n}]$ vs. k by simulation. For each worker, we sum s Pareto RVs samples. We then compare the n workers' service times to get $Y_{k:n}$. We estimate $\mathbb{E}[Y_{k:n}]$ by calculating the average of 10000 values of $Y_{k:n}$. We consider a system with $n = 12$ workers and four different values of the tail index $\alpha \in \{1.3, 2, 3, 5\}$. The simulation results are plotted in Fig. 9. We observe that splitting is optimal for a light tail (large α), and coding is optimal for a heavy tail (small α). When coding is optimal, the optimal code rate is close to 1/2. (Recall that we do not consider all fractions as possible code rates.)

In Fig. 10, we compare replication and its lower bound with splitting. The result for replication is from simulation, and the results for splitting and the lower bound are from their expressions. We observe that $\mathbb{E}[Y_{1:n}]$ with replication is much larger than its lower bound. Nevertheless, the lower

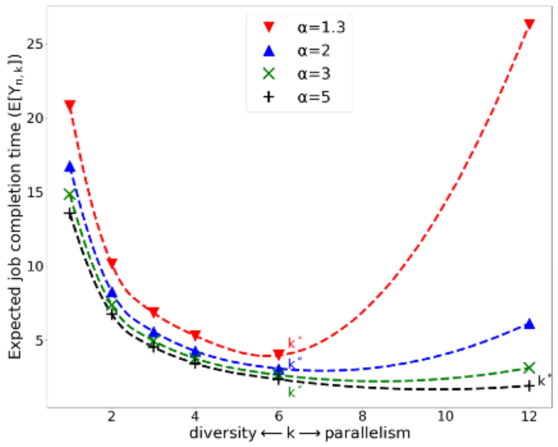

Fig. 9. Expected job completion time $\mathbb{E}[Y_{k:n}]$ for Pareto service time with additive scaling as a function of the diversity/parallelism parameter k. The number of workers (job size) is $n = 12$, and task size per worker is $s = n/k$ (Since both k and s are integers, we have $k \in \{1, 2, 3, 4, 6, 12\}$. Results are obtained by simulation. We use dashed curves to connect the points corresponding to different allowed values of k for a given α.) The Pareto scale parameter is $\lambda = 1$. Splitting or coding is optimal depending on the value of α.

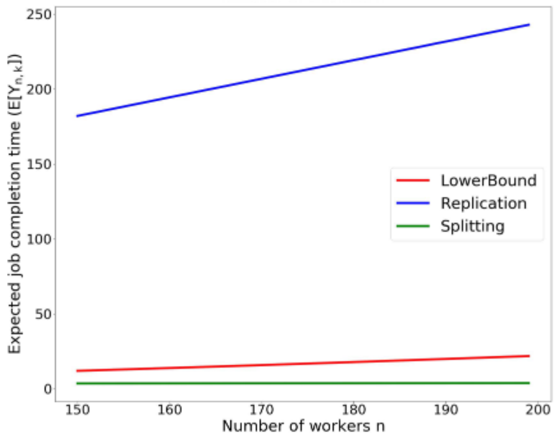

Fig. 10. Expected job completion time $\mathbb{E}[Y_{k:n}]$ for Pareto service time with additive scaling as a function of the number of workers (job size) n. The Pareto scale parameter is $\lambda = 1$ and the shape parameter is $\alpha = 4.5$. We set $\eta = 1$. Replication results are obtained by simulation. For Splitting and Lower Bound, we use (21) and (12), respectively. The lower bound on replication shows that splitting outperforms replication.

bound is clearly larger than $\mathbb{E}[Y_{n:n}]$ with splitting. Thus we conclude that splitting outperforms replication by achieving a lower expected job completion time for the large n scenario.

VI. Bi-Modal Service Time

Under the Bi-Modal model, the CU service time is given by X, where $X \sim$ Bi-Modal(B, ϵ) as given in (1). The expected job completion time $\mathbb{E}[Y_{k:n}]$ depends on the service time of $s = n/k$ CUs, which is determined by service time scaling model. In the following three subsections, we determine $\mathbb{E}[Y_{k:n}]$ for our three scaling models.

A. Server-Dependent Scaling

Under the server-dependent scaling, the service time of a task consisting of s CUs is given by $Y = s \cdot X$. Therefore, the job completion time is given by

$$Y_{k:n} = s \cdot X_{k:n}, \quad \text{where} \quad X \sim \text{Bi-Modal}(B, \epsilon)$$

It is easy to see that $X_{k:n}$ is a Bi-Modal random variable

$$X_{k:n} = \begin{cases} B & \text{w.p. } \sum_{i=0}^{k-1} \binom{n}{i}(1-\epsilon)^i \epsilon^{n-i} \\ 1 & \text{w.p. } 1 - \sum_{i=0}^{k-1} \binom{n}{i}(1-\epsilon)^i \epsilon^{n-i} \end{cases}$$

and that its expectation is given by

$$\mathbb{E}[Y_{k:n}] = s + s(B-1) \sum_{i=0}^{k-1} \binom{n}{i} \epsilon^{n-i}(1-\epsilon)^i. \tag{14}$$

From the definition of the Bi-Modal(B, ϵ) distribution given in (1), we see that when the probability of straggling ϵ is small ($\epsilon \to 0$), then X is highly concentrated around 1, and when ϵ is large ($\epsilon \to 1$), then X is highly concentrated around B. Therefore, in these two extreme cases, we have little variation in X, and thus diversity at the expense of parallelism does not help. Therefore, $\mathbb{E}[Y_{k:n}]$ reaches its minimum at $k = n$, which means that splitting is optimal.

Another case, in which we also have little variation in X, is when the magnitude of straggling B is small. We formally show that splitting is optimal in Proposition 1 for $B \le 2$.

Proposition 1: For Bi-Modal(B, ϵ) service time with server-dependent scaling, if $B \le 2$, the expected job completion time $\mathbb{E}[Y_{k:n}]$ reaches its minimum at $k = n$ (maximal parallelism).

Proof: Since k is an integer that divides n, we know that either $k = n$ or $k \le n/2$. When $k = n$, then $\mathbb{E}[Y_{n:n}] = 1 + (B-1)\sum_{i=0}^{n-1} \binom{n}{i} \epsilon^{n-i}(1-\epsilon)^i < B \le 2$. When $k \le n/2$, then $\mathbb{E}[Y_{k:n}] > s = n/k \ge 2$. □

When $B > 2$, computing k that minimizes the expression for $\mathbb{E}[Y_{k:n}]$ in (14) becomes harder, and we use the LLN to approximate $\mathbb{E}[Y_{k:n}]$ for large n, as follows:

Theorem 8: For the Bi-Modal(B, ϵ), where $B > 2$, service time with server-dependent scaling, we have

$$\mathbb{E}[Y_{k:n}] \sim \frac{1}{r} p_r + \frac{B}{r} q_r \quad \text{as} \quad n \to \infty \tag{15}$$

Here, $r = k/n$ is the code rate, $p_r \to 1$ if $1 - \epsilon > r$ and $p_r \to 0$ if $1 - \epsilon < r$, and $q_r = 1 - p_r$.

Proof: At server i, $i = 1, \dots, n$, the task completion time Y_i is a Bi-Modal random variable taking value $s \cdot 1$ or $s \cdot B$. Therefore, the job completion time $Y_{k:n}$ is also a Bi-Modal random variable taking value s or sB. We define an indicator function $\mathbf{1}_{\{Y_i|\{s\}\}} : \{Y_i|\{s, sB\}\} \to \{0, 1\}$, which takes value one when Y_i takes value s and zero otherwise. Let M be the sum of n i.i.d. indicators $\mathbf{1}_{\{Y_i|\{s\}\}}$ where $i = 1, \dots, n$. (M is the number of servers whose completion time took value s in a given realization.) Then $\Pr\{Y_{k:n} = s\} = \Pr\{M \ge k\}$ *and* $\Pr\{Y_{k:n} = sB\} = \Pr\{M < k\}$.

We next look into M through the LLN lens. Let $r \doteq k/n$ and $M_n \doteq M/n$. Observe that $M_n \to 1 - \epsilon$ as $n \to \infty$. We see that $p_r = \Pr\{Y_{k:n} = s\} = \Pr\{M_n \ge r\} \to 1$ if $1 - \epsilon > r$ and 0 if the inequality is reversed. Since $s = 1/r$, the LLN approximation yields the expression (15). □

From (15), we see that $\mathbb{E}[Y_{k:n}]$ is a convex, unimodal function of r on $[0, 1-\epsilon]$, and a decreasing function of r on $(1-\epsilon, 1]$. Therefore, $\mathbb{E}[Y_{k:n}]$ has two local minimums: $1/(1-\epsilon)$ at $r = 1-\epsilon$ and B at $r = 1$, which we compare and conclude the following. Whether coding or splitting is optimal depends on how the probability of straggling ϵ and straggling magnitude B compare to each other. When $\epsilon \le (B-1)/B$, the global minimum is $1/(1-\epsilon)$, and thus coding with the code rate $r = 1-\epsilon$ is optimal. When $\epsilon > (B-1)/B$, the global minimum is B, and thus splitting is optimal.

Notice that the LLN based approximation provides both qualitative and quantitative insights. We obtain a useful approximation to the optimal code rate in (15), which is much simpler and insightful than the exact expression (14). Moreover, from the numerical analysis below, we can observe that the approximation is useful even for small n.

Remark: Based on the insight we gained from the above analysis, we make the following conjecture about the optimal strategy for a general class of service time PDFs. The claim in the conjecture holds for the PDFs considered in this paper.

Conjecture 1: Under server-dependent scaling and a CU service time $X = \delta + Z$ where $\delta \ge 0$ is a constant and Z is a random variable whose support includes 0, the expected job completion time is minimized by replication when $\delta = 0$ and by coding or splitting otherwise.

Numerical Analysis: In Fig. 11, we evaluate the expression of $\mathbb{E}[Y_{k:n}]$ to see how the expected job completion time changes with the diversity/parallelism parameter k. We consider a system with $n = 12$ workers for six different values of $\epsilon \in \{0.005, 0.2, 0.4, 0.6, 0.8, 0.9\}$. Some observations can be made from the figure: when $\epsilon \to 0$ (e.g. 0.005), $\mathbb{E}[Y_{k:n}]$ decreases with k, and splitting is optimal. When ϵ is small (e.g. 0.2, 0.4, 0.6), $\mathbb{E}[Y_{k:n}]$ reaches its minimum at $k \in [2, 6]$, thus coding is optimal and the optimal code rate decreases with increasing ϵ. When ϵ is large (e.g. 0.8, 0.9), $\mathbb{E}[Y_{k:n}]$ reaches its minimum at $k = 12$, and splitting is optimal. From these observations, we conclude that as ϵ increases, $\mathbb{E}[Y_{k:n}]$ is minimized by introducing more diversity. However, when ϵ approaches 1, X approaches to a deterministic random variable, then $\mathbb{E}[Y_{k:n}]$ is minimized by maximal parallelism.

In Fig. 12, we evaluate $\mathbb{E}[Y_{k:n}]$ vs. k for four different values of B (from 2 to 15). We observe that when B is small (e.g., 2, 5), splitting is optimal. When B is large (e.g. 10, 15), $\mathbb{E}[Y_{k:n}]$ is minimized by coding and the optimal code rate increases with B. When B is very large (e.g. $B > 150$),

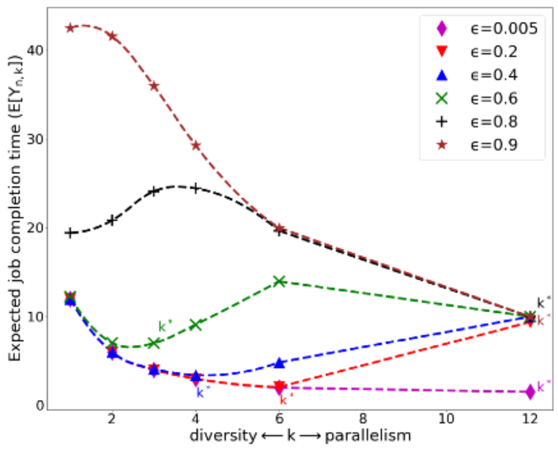

Fig. 11. Expected job completion time $\mathbb{E}[Y_{k:n}]$ for Bi-Modal service time with server-dependent scaling as a function of the diversity/parallelism parameter k (cf. (14)). The straggling magnitude B is 10. The number of workers (job size) is $n = 12$, and task size per worker is $s = n/k$ (Since both k and s are integers, we have $k \in \{1, 2, 3, 4, 6, 12\}$. We use dashed curves to connect the points corresponding to different allowed values of k for a given α.) The optimal code rate decreases as ϵ increases, except when ϵ approaches 1, where the maximal parallelism is optimal.

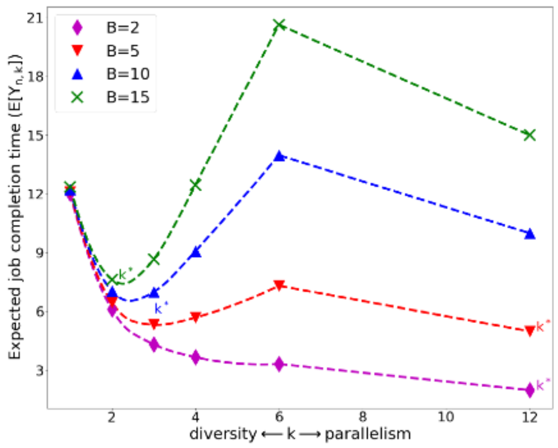

Fig. 12. Expected job completion time $\mathbb{E}[Y_{k:n}]$ for Bi-Modal service time with server-dependent scaling as a function of the diversity/parallelism parameter k (cf. (14)). The straggling probability ϵ is 0.6. The number of workers (also job size) is $n = 12$, and task size per worker is $s = n/k$ (Since both k and s are integers, we have $k \in \{1, 2, 3, 4, 6, 12\}$. We use dashed curves to connect the points corresponding to different allowed values of k for a given α.) The optimal code rate decreases with increasing B.

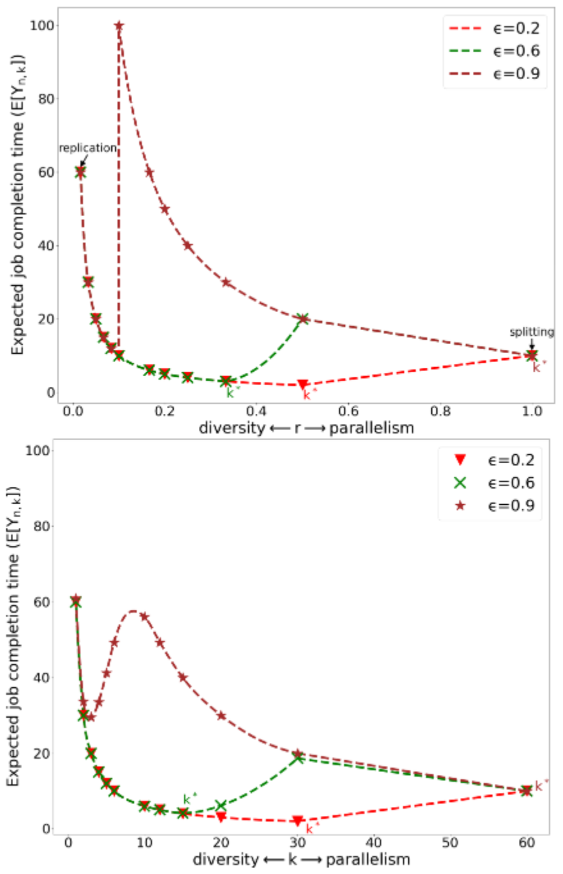

Fig. 13. Comparisons between the LLN approximation of $\mathbb{E}[Y_{k:n}]$ given by (15) and the exact result given by (14) for Bi-Modal service time with server-dependent scaling. The straggling magnitude B is 10. The number of workers (job size) is $n = 60$, task size per worker is $s = n/k$. (upper) The LLN approximation shows the $\mathbb{E}[Y_{k:n}]$ vs. the code rate $r = k/n$. (lower) The exact dependence of $\mathbb{E}[Y_{k:n}]$ on the diversity/parallelism parameter k. (Since both k and s are integers, we have $k \in \{1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30, 60\}$. We use dashed curves to connect the points corresponding to different allowed values of k for a given α.) The LLN approximation performs well here.

not shown in Fig. 12, replication is optimal. From the above, we conclude that the magnitude of B determines the diversity/parallelism trade-off: when B is small, we gain more from parallelism, s.t. splitting. When B is large, we gain more from diversity, s.t. coding or replication.

In Fig. 13, we compare the LLN approximation of $\mathbb{E}[Y_{k:n}]$ with the exact result (14). The upper graph shows the LLN approximation of the dependence of $\mathbb{E}[Y_{k:n}]$ on $r = k/n$, and the lower graph shows the exact dependence of $\mathbb{E}[Y_{k:n}]$ on k. There are $n = 60$ workers and three different values of ϵ. Since k and s are integers, we can only have $k \in \{1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30, 60\}$. The corresponding $\mathbb{E}[Y_{k:n}]$ are marked in the figure. To evaluate the approximation, we compare three important metrics: the local minimums, the optimal k^* and the minimum $\mathbb{E}[Y_{k:n}]$, and make the following observations. First, for each value of ϵ, both LLN and the exact result have the same number of local minimums. However, when $\epsilon = 0.9$, the LLN gives a much smaller value of the first local minimum. Second, when $\epsilon = 0.2$ and 0.9, the LLN shows the same values of k^*'s as the exact result. When $\epsilon = 0.6$, the LLN approximate optimal value is $r = 1/3$ ($k^* = 20$), whereas the exact value is $k^* = 15$. Third, the minimum $\mathbb{E}[Y_{k:n}]$'s in the LLN are close to the values in the exact result. Therefore, we see that in spite of some differences, the LLN approximation is good, as it shows well the general trend and gives some exact results. Furthermore, the approximation should be even better for larger n.

B. Data-Dependent Scaling

Under the data-dependent scaling, the service time of a task consisting of s CUs is given by $Y = s \cdot \Delta + X$. Therefore, the job completion time is given by

$$Y_{k:n} = s \cdot \Delta + X_{k:n}, \quad \text{where} \quad X \sim \text{Bi-Modal}(B, \epsilon)$$

and, by using the expression for $\mathbb{E}[X_{k:n}]$ in (14), we have

$$\mathbb{E}[Y_{k:n}] = s\Delta + 1 + (B-1)\sum_{i=0}^{k-1}\binom{n}{i}\epsilon^{n-i}(1-\epsilon)^{i} \qquad (16)$$

Note that $s\Delta$ is a decreasing function of k (since $s = n/k$), whereas $\mathbb{E}[X_{k:n}]$ is an increasing function of k (by the definition of order statistics). Then, there is a balance between $s\Delta$ and $\mathbb{E}[X_{k:n}]$ that minimizes the expected job completion time $\mathbb{E}[Y_{k:n}]$. However, since the expression of $\mathbb{E}[Y_{k:n}]$ is very complicated, it is difficult to find the minimum value. Instead of finding the exact value of minimum $\mathbb{E}[Y_{k:n}]$, we can find the approximation by applying law of large numbers for the large n scenario.

Large n Scenario:

By applying LLN, we find the approximation for the expected job completion time $\mathbb{E}[Y_{k:n}]$ in Theorem 9.

Theorem 9: Considering Bi-Modal(B, ϵ) service time with data-dependent scaling, when n is sufficiently large, we find the LLN approximation for $\mathbb{E}[Y_{k:n}]$,

$$\mathbb{E}[Y_{k:n}] \sim \frac{\Delta}{r} + p_r + Bq_r, \quad \text{as} \quad n \to \infty \qquad (17)$$

where $r = \frac{k}{n}$ is the code rate, $p_r \to 1$ if $1 - \epsilon > r$ and 0 if the inequality is reversed, and $q_r = 1 - p_r$.

Proof: At server i, $i = 1, \ldots, n$, the task completion time $Y_i = s\Delta + X_i$, where X_i is a Bi-Modal random variable taking value 1 or B. Therefore, the job completion time $Y_{k:n} = s \cdot \Delta + X_{k:n}$. We define an indicator function $\mathbf{1}_{\{X_i|\{1\}\}} : \{X_i|\{1, B\}\} \to \{0, 1\}$. Let M be the sum of n i.i.d. indicators $\mathbf{1}_{\{X_i|\{1\}\}}$ where $i \in \{1, \cdots, n\}$. Then $\Pr\{X_{k:n} = B\} = \Pr\{M < k\}$ and $\Pr\{X_{k:n} = 1\} = \Pr\{M \geq k\}$

We next look into M through the LLN lens. Let $r \doteq \frac{k}{n}$ and $M_n \doteq M/n$. Observe that $M_n \to 1 - \epsilon$ as $n \to \infty$. From the above, we see that $p_r = \Pr\{X_{k:n} = 1\} = \Pr\{M_n \geq r\} \to 1$ if $1-\epsilon > r$ and 0 if the inequality is reversed, and $q_r = 1 - p_r$. Since $Y_{k:n} = s\cdot\Delta + X_{k:n}$ and $s = 1/r$, the LLN approximation yields the expression (17). □

From (17), we see that $\mathbb{E}[Y_{k:n}]$ is a convex, unimodal function of r on $[0, 1-\epsilon]$ and a decreasing function of r on $(1-\epsilon, 1]$. Therefore, $\mathbb{E}[Y_{k:n}]$ has two local minimums: $1 + \Delta/(1-\epsilon)$ at $r = 1 - \epsilon$ and $\Delta + B$ at $r = 1$, which we compare and reach the following conclusions. When $\epsilon \leq (B-1)/(\Delta + B - 1)$, the global minimum is $1 + \Delta/(1-\epsilon)$, and thus coding at rate $r = 1 - \epsilon$ is optimal. When $\epsilon > (B-1)/(\Delta + B - 1)$, the global minimum is $\Delta + B$, and thus splitting is optimal.

Numerical Analysis: In Fig. 14, we evaluate $\mathbb{E}[Y_{k:n}]$ vs. k. There are $n = 12$ workers and $\Delta = 5$. By comparing five different values of $\epsilon \in \{0.05, 0.2, 0.5, 0.6, 0.9\}$, we observe that when $\epsilon \to 0$ (e.g. 0.05), $\mathbb{E}[Y_{k:n}]$ decreases with increasing k, thus splitting is optimal. When ϵ is small (e.g. 0.2, 0.5), $\mathbb{E}[Y_{k:n}]$ reaches its minimum at $k = 6$, thus coding is optimal. When ϵ is large (e.g. 0.6, 0.9), $\mathbb{E}[Y_{k:n}]$ decreases with increasing k again, thus splitting is optimal. From these observations, we conclude that as ϵ increases, $\mathbb{E}[Y_{k:n}]$ is minimized by introducing more diversity, but when ϵ approaches 1, maximal parallelism is optimal.

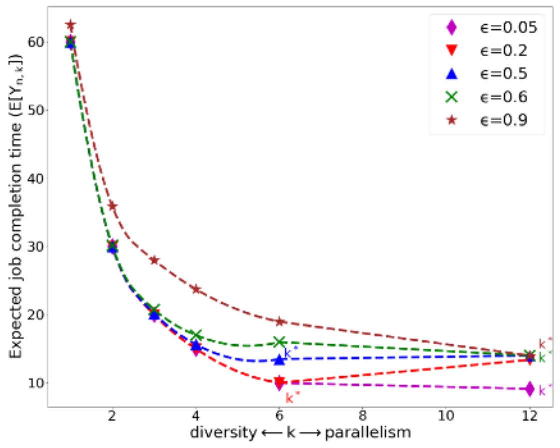

Fig. 14. Expected job completion time $\mathbb{E}[Y_{k:n}]$ for Bi-Modal service time with data-dependent scaling as a function of the diversity/parallelism parameter k (cf. (16)). The straggling magnitude B is 10. The number of workers (job size) is $n = 12$, and task size per worker is $s = n/k$ (Since both k and s are integers, we have $k \in \{1, 2, 3, 4, 6, 12\}$. We use dashed curves to connect the points corresponding to different allowed values of k for a given α.) When ϵ is small, the optimal code rate decreases with increasing ϵ; When ϵ is relatively large, maximal parallelism is optimal.

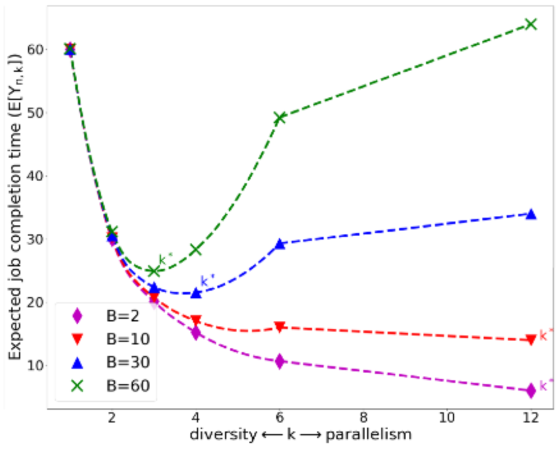

Fig. 15. Expected job completion time $\mathbb{E}[Y_{k:n}]$ for Bi-Modal service time with data-dependent scaling as a function of the diversity/parallelism parameter k (cf. (16)). The straggling probability ϵ is 0.6. The number of workers (job size) is $n = 12$, and task size per worker is $s = n/k$ (Since both k and s are integers, we have $k \in \{1, 2, 3, 4, 6, 12\}$. We use dashed curves to connect the points corresponding to different allowed values of k for a given α.) The optimal code rate decreases with increasing B.

In Fig. 15, we analyze $\mathbb{E}[Y_{k:n}]$ vs. k in a system with $n = 12$ workers and $\Delta = 5$ for four different values of $B \in \{2, 10, 30, 60\}$. The diversity/parallelism trade-off is determined by the magnitude of B. When B is small (e.g. 2, 10), $\mathbb{E}[Y_{k:n}]$ decreases with increasing k, thus splitting is optimal. When B is large (e.g. 30, 60), $\mathbb{E}[Y_{k:n}]$ reaches its minimum at $k = 6$, thus coding is optimal. Notice that if $\Delta = 0$, replication is optimal; If $\Delta \gg B$, splitting is optimal.

In Fig. 16, we compare the LLN approximation of $\mathbb{E}[Y_{k:n}]$ (upper) with the exact result (16) (lower). There are $n = 60$ workers, $\Delta = 5$, and three values of ϵ. Since $\mathbb{E}[Y_{k:n}]$ is very

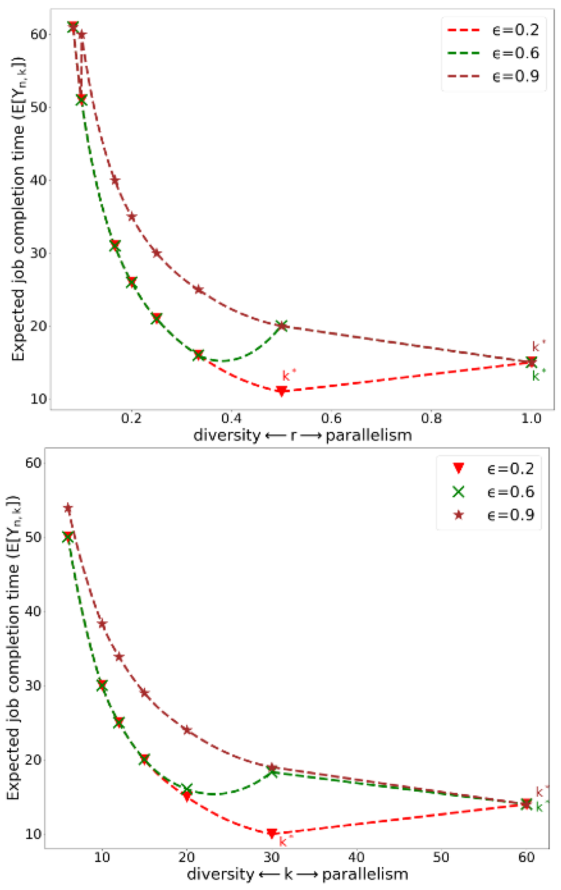

Fig. 16. Comparisons between the LLN approximation of $\mathbb{E}[Y_{k:n}]$ given by (17) and the exact result given by (16) for Bi-Modal service time with data-dependent scaling. The straggling magnitude B is 10. The number of workers (job size) is $n = 60$, task size per worker is $s = n/k$. (upper) The LLN approximation shows the $\mathbb{E}[Y_{k:n}]$ vs. the code rate $r = k/n$. (lower) The exact dependence of $\mathbb{E}[Y_{k:n}]$ on the diversity/parallelism parameter k. (Since both k and s are integers, we have $k \in \{1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30, 60\}$. We use dashed curves to connect the points corresponding to different allowed values of k for a given α.) The LLN approximation performs well here.

large when k (r) is small, we only plot the points for $k \geq 5$ ($r > 1/12$). We compare three important metrics to evaluate the approximation: the local minimums, the optimal k^* and the minimum $\mathbb{E}[Y_{k:n}]$, and observe the following. First, for each value of epsilon, both the LLN and the exact result have the same number of local minimums. The values of local minimums in both graphs are close to each other. Second, the LLN shows the same values of k^*'s as the exact result. Third, the minimum $\mathbb{E}[Y_{k:n}]$'s obtained by the LLN approximation are close to the exact values. Overall, the LLN gives a very good approximation to the exact result, and the approximation will be more even more accurate when n is larger.

C. Additive Scaling

Under the additive scaling, the service time of a task consisting of s CUs is given by

$$Y = X_1 + \cdots + X_s, \quad \text{where } X_i \sim \text{Bi-Modal}(B, \epsilon).$$

We derive the expressions for Y and the expected job completion time $\mathbb{E}[Y_{k:n}]$ in Lemma 1 (see Appendix A-4). These

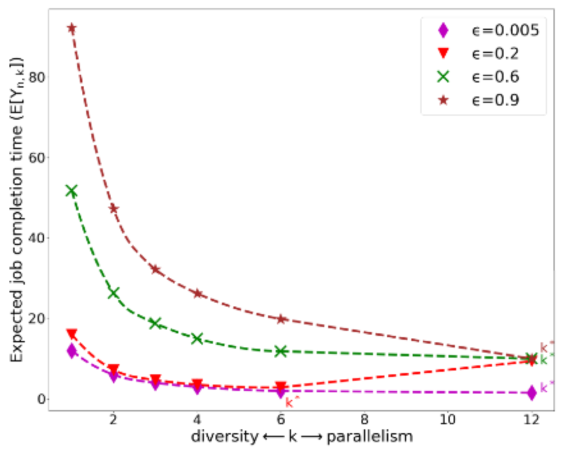

Fig. 17. Expected job completion time $\mathbb{E}[Y_{k:n}]$ for Bi-Modal service time with additive scaling as a function of the diversity/parallelism parameter k (cf. (24)). The straggling magnitude B is 10. The number of workers (job size) is $n = 12$, and task size per worker is $s = n/k$ (Since both k and s are integers, we have $k \in \{1, 2, 3, 4, 6, 12\}$. We use dashed curves to connect the points corresponding to different allowed values of k for a given α.) When ϵ approaches to 0 or 1, maximal parallelism is optimal. Otherwise, coding is optimal, and the code rate is around $1/2$.

expressions are unsuitable for theoretical analysis, but can be numerically evaluated. By theoretical analysis, we only find that splitting is optimal when $B \leq 2$ in Proposition 2.

Proposition 2: For Bi-Modal(B, ϵ) service time, if $B \leq 2$, the expected job completion time $\mathbb{E}[Y_{k:n}]$ reaches its minimum when $k = n$ (maximal parallelism).

Proof: When $k = n$, we have $s = n/k = 1$, and thus $Y \sim$ Bi-Modal(B, ϵ). Then we have $Y_{n:n} = 1$ with the probability $(1-\epsilon)^n$, and $Y_{n:n} = B$ with the probability $1 - (1-\epsilon)^n$. Therefore, $\mathbb{E}[Y_{n:n}] \leq B \leq 2$. When $k < n$, we have $s = n/k \geq 2$, then $\mathbb{E}[Y_{k:n}] > 2$. □

Numerical Analysis: In Fig. 17, we evaluate $\mathbb{E}[Y_{k:n}]$ vs. k. We consider a system with $n = 12$ workers for four different values of $\epsilon \in \{0.005, 0.2, 0.6, 0.9\}$. Some observations can be made from the figure: when $\epsilon \to 0$ (e.g. 0.005), splitting outperforms the other two strategies slightly. When ϵ is small (e.g. 0.2), there is a balance between diversity and parallelism, and coding with the code rate $1/2$ is optimal. When ϵ is large (e.g. 0.6, 0.9), splitting is optimal. These observations are similar to those for server-dependent and data-dependent scaling. We conjecture that coding with a proper code rate is always better than replication in Conjecture 2. Recall that this is not the case for server-dependent and data-dependent scaling, where replication may be optimal for certain (large) values of B.

Conjecture 2: For Bi-Modal(B, ϵ) service time with additive scaling, either coding or splitting outperforms replication, i.e. there exists k, $2 \leq k \leq n$ such that $\mathbb{E}[Y_{k:n}] < \mathbb{E}[Y_{1:n}]$.

In Fig. 18, we plot $\mathbb{E}[Y_{k:n}]$ vs. k in a system with $n = 12$ workers for four different values of $B \in \{2, 5, 10, 20\}$. We see that the diversity/parallelism trade-off is determined by the magnitude of straggling B. The figure also shows that the optimal code rate is either $1/2$ or 1, which coincides with Conjecture 2. To examine this result, we evaluated $\mathbb{E}[Y_{k:n}]$ for values of B from 2 to 10000. We observed that when

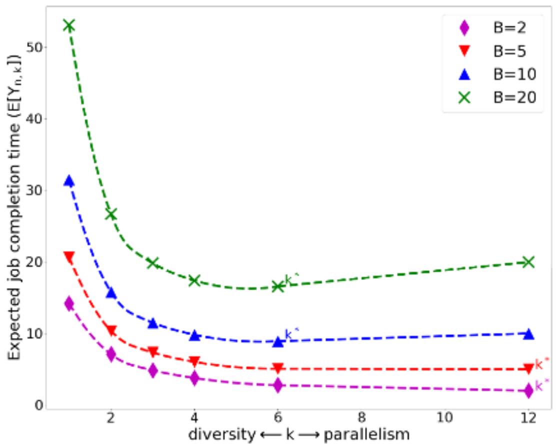

Fig. 18. Expected job completion time $\mathbb{E}[Y_{k:n}]$ for Bi-Modal service time with additive scaling as a function of the diversity/parallelism parameter k (cf. (24)). The straggling probability ϵ is 0.4. The number of workers (job size) is $n = 12$, and task size per worker is $s = n/k$ (Since both k and s are integers, we have $k \in \{1, 2, 3, 4, 6, 12\}$. We use dashed curves to connect the points corresponding to different allowed values of k for a given α.) The optimal code rate decreases as B increasing, reaching the minimum at $1/3$.

$B \leq 106$, the optimal code rate is $1/2$, and when $B \geq 107$, the optimal code rate is $1/3$. These simulation results provide some support to Conjecture 2.

Remark: Based on the insight we gained from the analysis up to this point, we make the following conjecture about the optimal strategy for general service time PDFs. The claim in the conjecture holds for the PDFs considered in this paper.

Conjecture 3: Under additive scaling and a general CU service time, either coding or splitting outperforms replication.

VII. Conclusion and Future Directions

In distributed computing with redundancy, smaller tasks that make up a large computing job are executed in parallel, and redundancy is added as a form of diversification that reduces the job service time dependence on the execution of straggling tasks. Both parallelism and diversity reduce job completion time. However, in systems where a constant number of workers is available for job execution, more redundancy means less parallelism and vice versa. We considered the trade-off between diversity and parallelism with the purpose to minimize the job completion time.

Depending on the level of redundancy used in the system, each worker has to execute a task consisting of one or more computing units (a minimum-size task below which distributed computing would be inefficient). In Sections IV, V, and VI, we considered three common models for the computing unit service time PDF. For each of these models, we adopted three common assumptions about service time scaling with the task size. For each service time model, the results are summarised in a table at the end of the corresponding section. We also drew several conclusions and conjectures about a general service time distributions with the finite fourth moment.

In our summary of the results in Table I, we distinguished only between the following three regimes: 1) maximum parallelism (splitting the job across the workers), 2) maximum diversity (replication of the job at each worker), and 3) the region where coding is used to enable a trade-off between diversity and parallelism. We indicated in the table how the optimal strategy changes as the tail of the service time PDF becomes heavier. The general conclusion is that the optimal level of redundancy strongly depends on the assumptions made about the task service time PDF and its scaling with the task size. This work sets the stage for many problems of interest to be studied in the future. We briefly describe three directions of immediate interest.

1) Diversity/Parallelism Tradeoff for Non-MDS Codes: As we mentioned in Sec. II-C, our system model and analysis approach are not limited to MDS codes. If an $[n, k]$ code with minimum distance d is used, then the job is completed when any $m = n - (d - 1)$ out of n tasks are completed. The Singleton bound imposes the constraint $m \geq k$, where $m = k$ for MDS codes. When $m > k$, the diversity/parallelism tradeoff may change, since for such systems, the task size $s = n/k$ is the same as the MDS coded ones, but they are able to mitigate fewer stragglers. Of particular interest is characterizing the diversity/parallelism tradeoff for codes with features that are attractive in practice, e.g., the codes proposed in [23] with linear encoding and decoding. Observe that characterizing the diversity/parallelism tradeoff is a complementary task to code design and selection. It determines the code parameters within a class of codes that are optimal under the given system model.

2) Diversity/Parallelism Tradeoff for General Job Sizes: Since we are concerned with systems with a fixed number of workers n, we have assumed that each job is split into n CUs (has size n). To generalize this assumption, we can set the job size to be bn CUs, where b is an integer and $b \geq 2$. This generalization will require that some of our results be slightly modified (Theorem. 1, 6 and 8). Some other claims may need new statements or proofs, e.g., Theorem. 4 and 7. This generalization will allow us to analyze the full spectrum of code rates. Further generalizations of job sizes may be much more difficult but worth studying.

3) Diversity/Parallelism Tradeoff for Other System Models: This paper analyzed the most common service time and scaling models. Some other distributions, e.g., Weibull distribution used in [14] are also of practical interest. In previous sections, we stated Conjectures 1, 2 and 3 regarding server-dependent and additive scaling. Conjectures 1 and 3 concern general service time distributions, but knowing whether they hold for classes of distributions other than those considered in the paper is of interest. This paper considered distributed, parallel architectures commonly implemented in modern computing frameworks, e.g., Kubernetes and Apache Mesos, and adopted the model that corresponds to these systems. However, other computing architectures which give rise to different models are also important to study. One such example is the emerging wireless edge computing where, for example, the worker nodes may not be statistically identical.

Appendix
Mathematical Background

A. Order Statistics

In executing jobs with redundant tasks, the notion of order statistics plays a central role. We here state the results we use throughout the paper. More information can be found in, e.g., [68]–[71]. Let $X_1, X_2, \ldots, X_n$ be n samples of some RV X. Then the k-th smallest is an RV denoted by $X_{k:n}$ and known as the k-th order statistic of $X_1, X_2, \cdots, X_n$.

1) Exponential Distribution: If $X_1, X_2, \cdots, X_n$ are $\text{Exp}(W)$, then $X_{1:n}$ is $\text{Exp}(W/n)$, and

$$X_{k:n} = X_{1:n} + X_{1:(n-1)} + \cdots + X_{1:(n-k+1)}. \tag{18}$$

The expectation of $X_{k:n}$ is given by

$$\mathbb{E}[X_{k:n}] = W \sum_{i=1}^{k} \frac{1}{n-k+i} = W\big(H_n - H_{n-k}\big) \tag{19}$$

where H_n is (generalized) harmonic numbers defined as $H_n = \sum_{j=1}^{n} \frac{1}{j}$. We often use the approximation $H_n = \log n + \gamma + \mathcal{O}(n^{-1})$, where $\gamma = 0.577$ is Euler's constant.

2) Erlang Distribution: If $X_1, X_2, \cdots, X_n$ are $\text{Erlang}(s, W)$, then, according to the formula of gamma order statistics in [72], we have

$$\begin{aligned} X_{k:n} = &\frac{Wk}{(s-1)!}\binom{n}{k}\sum_{i=0}^{k-1}(-1)^i\binom{k-1}{i} \\ &\sum_{j=0}^{(s-1)(n-k+i)} \alpha_j(s, n-k+i)\frac{(s+j)!}{(n-k+i+1)^{s+j+1}} \end{aligned} \tag{20}$$

where $\alpha_z(x, y)$ is the coefficient of t^z in the expansion of $\big(\sum_{l=0}^{x-1} t^l/l!\big)^y$.

3) Pareto Distribution: If $X_1, X_2, \cdots, X_n$ are $\text{Pareto}(\lambda, \alpha)$, then the expectation of $X_{k:n}$ for $\alpha > 1$ is given by

$$\mathbb{E}[X_{k:n}] = \lambda \frac{n!}{(n-k)!}\frac{\Gamma(n-k+1-1/\alpha)}{\Gamma(n+1-1/\alpha)} \tag{21}$$

where the complete gamma function is defined as $\Gamma(x) = \int_0^\infty u^{x-1}e^{-u}du$.

In our work on straggler mitigation (see [41]), we have obtained the following approximation:

$$\Gamma(x+\beta)/\Gamma(x+\alpha) \sim x^{\beta-\alpha} \tag{22}$$

by using Stirling's approximation or by an induction on Gautschi's inequality [73]. This result is useful in finding numerically good approximations of $\mathbb{E}[X_{k:n}]$ for large n.

4) Bi-Modal Distribution:

Lemma 1: If $Y = X_1 + \cdots + X_s$ is the sum of s i.i.d. $\text{Bi-Modal}(B, \epsilon)$ RVs, then

$$Y = s - w + wB \;\text{ w.p. }\; \tbinom{s}{w}(1-\epsilon)^{s-w}\epsilon^w, \;\; 0 \le w \le s. \tag{23}$$

The expectation of $Y_{k:n}$ is

$$\begin{aligned} \mathbb{E}[Y_{k:n}] = s + (B-1)&\sum_{w=1}^{s-1} w \sum_{i=0}^{k-1}\binom{n}{i}\Big(\sum_{j=0}^{w-1} p_j\Big)^i \\ &\Big[\sum_{l=k-i}^{n-i}\binom{n-i}{l}p_w^l\Big(\sum_{h=w+1}^{s} p_h\Big)^{n-i-l}\Big] \\ &+ s(B-1)\sum_{i=0}^{k-1}\binom{n}{i}p_s^{n-i}(1-p_s)^i \end{aligned} \tag{24}$$

Proof: The expression in (23) is straightforward to derive. Since $Y_1, \cdots, Y_n$ are sums of s i.i.d. $\text{Bi-Modal}(B, \epsilon)$ RVs, we deduce $Y_{k:n}$ by using the definition of order statistics,

$$Y_{k:n} = \begin{cases} s & \text{w.p. } \sum_{i=0}^{n-k}\binom{n}{i}p_0^{n-i}(1-p_0)^i \\ s - w + wB & \text{w.p. } \Pr(w), 1 \le w < s \\ sB & \text{w.p. } \sum_{i=0}^{k-1}\binom{n}{i}p_s^{n-i}(1-p_s)^i \end{cases}$$

where $p_v = \binom{s}{v}(1-\epsilon)^{s-v}\epsilon^v$ for $v = 0, \ldots, s$. When $w = 0$ and $w = s$, the respective probabilities of $Y_{k:n} = s$ and $Y_{k:n} = sB$ are straightforward to derive. We consider the case when $0 < w < s$ and $Y_{k:n} = s - w + wB$. Since we are concerned with k-th order statistics, we know that there are at most $k-1$ RVs among $\{Y_1, \cdots, Y_n\}$ whose values are smaller than $s - w + wB$. Consider the event that i ($i \le k-1$) of the RVs are smaller than $s - w + wB$. The probability of this event is $\binom{n}{i}\big(\sum_{j=0}^{w-1} p_j\big)^i$. Among the remaining $n - i$ RVs in $\{Y_1, \cdots, Y_n\}$, there are at most $n - k$ RVs whose values are larger than $s - w + wB$; the others are equal to $s - w + wB$. Consider an event where ℓ ($k - i \le \ell \le n - i$) of these RVs take values larger than $s - w + wB$. Thus all other $n - i - \ell$ RVs take value $s - w + wB$. The probability of this event under the condition of the previous event is $\binom{n-i}{l}p_w^l\big(\sum_{h=w+1}^{s} p_h\big)^{n-i-l}$. Thus,

$$\begin{aligned} \Pr(w) = &\sum_{i=0}^{k-1}\binom{n}{i}\Big(\sum_{j=0}^{w-1} p_j\Big)^i \\ &\Big[\sum_{l=k-i}^{n-i}\binom{n-i}{l}p_w^l\Big(\sum_{h=w+1}^{s} p_h\Big)^{n-i-l}\Big]. \end{aligned}$$

Therefore, we have

$$\begin{aligned} \mathbb{E}[Y_{k:n}] = s + (B-1)&\sum_{w=1}^{s-1} w \sum_{i=0}^{k-1}\binom{n}{i}\Big(\sum_{j=0}^{w-1} p_j\Big)^i \\ &\Big[\sum_{l=k-i}^{n-i}\binom{n-i}{l}p_w^l\Big(\sum_{h=w+1}^{s} p_h\Big)^{n-i-l}\Big] \\ &+ s(B-1)\sum_{i=0}^{k-1}\binom{n}{i}p_s^{n-i}(1-p_s)^i \end{aligned}$$

□

B. A Generalized Birthday Problem

A generalized birthday problem is stated as follows: "How many draws with replacement on average have to be made from a set of n coupons until one of the coupons is drawn d times?" The expected number of draws $\mathbb{E}(n, d)$ was determined in [74]:

$$\mathbb{E}(n, d) = \int_0^\infty e^{-t}\Big[S_d\Big(\frac{t}{n}\Big)\Big]^n dt$$

$$\text{where } \; S_d(x) = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \cdots + \frac{x^{d-1}}{(d-1)!}. \tag{25}$$

We also use an asymptotic expression for (25) given in [74], which says that for a fixed d, we have

$$\mathbb{E}(n, d) \sim \sqrt[d]{d!}\,\Gamma(1 + 1/d)n^{1-\frac{1}{d}}, \;\; \text{as } \; n \to \infty. \tag{26}$$

References

[1] P. Peng, E. Soljanin, and P. Whiting, "Diversity vs. parallelism in distributed computing with redundancy," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2020, pp. 257–262.

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

[3] M. F. Aktaş and E. Soljanin, "Straggler mitigation at scale," *IEEE/ACM Trans. Netw.*, vol. 27, no. 6, pp. 2266–2279, Dec. 2019.

[4] K. Gardner, S. Zbarsky, S. Doroudi, M. Harchol-Balter, and E. Hyytia, "Reducing latency via redundant requests: Exact analysis," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 43, no. 1, pp. 347–360, 2015.

[5] G. Ananthanarayanan, A. Ghodsi, S. Shenker, and I. Stoica, "Effective straggler mitigation: Attack of the clones," in *Proc. 10th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2013, pp. 185–198.

[6] S. Kadhe, E. Soljanin, and A. Sprintson, "When do the availability codes make the stored data more available?" in *Proc. 53rd Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2015, pp. 956–963.

[7] S. Kadhe, E. Soljanin, and A. Sprintson, "Analyzing the download time of availability codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2015, pp. 1467–1471.

[8] W. Halbawi, N. Azizan, F. Salehi, and B. Hassibi, "Improving distributed gradient descent using Reed–Solomon codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 2027–2031.

[9] M. F. Aktas, S. Kadhe, E. Soljanin, and A. Sprintson, "Download time analysis for distributed storage codes with locality and availability," *IEEE Trans. Commun.*, vol. 69, no. 6, pp. 3898–3910, Jun. 2021.

[10] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover, "On the optimal recovery threshold of coded matrix multiplication," *IEEE Trans. Inf. Theory*, vol. 66, no. 1, pp. 278–301, Jan. 2020.

[11] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Coded MapReduce," in *Proc. 53rd Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2015, pp. 964–971.

[12] Y. Yang, P. Grover, and S. Kar, "Coded distributed computing for inverse problems," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 709–719.

[13] R. Yuster and U. Zwick, "Fast sparse matrix multiplication," *ACM Trans. Algorithms*, vol. 1, no. 1, pp. 2–13, 2005.

[14] A. Reisizadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr, "Coded computation over heterogeneous clusters," *IEEE Trans. Inf. Theory*, vol. 65, no. 7, pp. 4227–4242, Jul. 2019.

[15] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3368–3376.

[16] D. Wang, G. Joshi, and G. Wornell, "Using straggler replication to reduce latency in large-scale parallel computing," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 43, no. 3, pp. 7–11, Nov. 2015.

[17] G. Joshi, E. Soljanin, and G. Wornell, "Efficient redundancy techniques for latency reduction in cloud systems," *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 2, no. 2, pp. 1–30, 2017.

[18] M. Primorac, K. Argyraki, and E. Bugnion, "When to hedge in interactive services," in *Proc. Symp. Netw. Syst. Design Implement.*, 2021, pp. 373–387.

[19] S. Dutta, V. Cadambe, and P. Grover, "Short-dot: Computing large linear transforms distributedly using coded short dot products," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 2100–2108.

[20] 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.

[21] S. Dutta, V. Cadambe, and P. Grover, "Coded convolution for parallel and distributed computing within a deadline," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 2403–2407.

[22] Q. Yu, M. Maddah-Ali, and S. Avestimehr, "Polynomial codes: An optimal design for high-dimensional coded matrix multiplication," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4403–4413.

[23] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," *IEEE Trans. Inf. Theory*, vol. 66, no. 3, pp. 1920–1933, Mar. 2020.

[24] D. Merrill and M. Garland, "Merge-based sparse matrix-vector multiplication (SpMV) using the CSR storage format," *ACM SIGPLAN Notices*, vol. 51, no. 8, pp. 1–2, Nov. 2016.

[25] C. Karakus, Y. Sun, S. Diggavi, and W. Yin, "Redundancy techniques for straggler mitigation in distributed optimization and learning," *J. Mach. Learn. Res.*, vol. 20, no. 72, pp. 1–47, 2019.

[26] S. Kiani, N. Ferdinand, and S. C. Draper, "Exploitation of stragglers in coded computation," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 1988–1992.

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

[28] N. Raviv, Y. Cassuto, R. Cohen, and M. Schwartz, "Erasure correction of scalar codes in the presence of stragglers," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 1983–1987.

[29] M. Ye and E. Abbe, "Communication-computation efficient gradient coding," 2018, *arXiv:1802.03475*.

[30] N. Ferdinand and S. C. Draper, "Anytime stochastic gradient descent: A time to hear from all the workers," in *Proc. 56th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Oct. 2018, pp. 552–559.

[31] Y. Chen, A. S. Ganapathi, R. Griffith, and R. H. Katz, "Analysis and lessons from a publicly available Google cluster trace," Dept. EECS, Univ. California, Berkeley, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2010-95, 2010, vol. 94.

[32] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Towards understanding heterogeneous clouds at scale: Google trace analysis," Intel Sci. Technol. Center Cloud Comput., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. ISTC-CC-TR-12-101, 2012, vol. 84.

[33] G. Ananthanarayanan, M. C.-C. Hung, X. Ren, I. Stoica, A. Wierman, and M. Yu, "GRASS: Trimming stragglers in approximation analytics," in *Proc. 11th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2014, pp. 289–302.

[34] K. Gardner, M. Harchol-Balter, A. Scheller-Wolf, and B. Van Houdt, "A better model for job redundancy: Decoupling server slowdown and job size," *IEEE/ACM Trans. Netw.*, vol. 25, no. 6, pp. 3353–3367, Dec. 2017.

[35] M. F. Aktas and E. Soljanin, "Optimizing redundancy levels in master-worker compute clusters for straggler mitigation," 2019, *arXiv:1906.05345*.

[36] G. Joshi, Y. Liu, and E. Soljanin, "Coding for fast content download," in *Proc. 50th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Oct. 2012, pp. 326–333.

[37] G. Joshi, Y. Liu, and E. Soljanin, "On the delay-storage trade-off in content download from coded distributed storage systems," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 989–997, May 2014.

[38] K. Psounis, P. Molinero-Fernández, B. Prabhakar, and F. Papadopoulos, "Systems with multiple servers under heavy-tailed workloads," *Perform. Eval.*, vol. 62, nos. 1–4, pp. 456–474, Oct. 2005.

[39] N. Bansal and M. Harchol-Balter, "Analysis of SRPT scheduling: Investigating unfairness," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 29, no. 1, pp. 279–290, 2001.

[40] A. Vulimiri, P. B. Godfrey, R. Mittal, J. Sherry, S. Ratnasamy, and S. Shenker, "Low latency via redundancy," in *Proc. 9th ACM Conf. Emerg. Netw. Exp. Technol.*, Dec. 2013, pp. 283–294.

[41] M. F. Aktas, P. Peng, and E. Soljanin, "Effective straggler mitigation: Which clones should attack and when?" *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 45, no. 2, pp. 12–14, 2017.

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

[43] G. Joshi, "Synergy via redundancy: Boosting service capacity with adaptive replication," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 45, no. 3, pp. 21–28, Mar. 2018.

[44] S. Kwon and N. Gautam, "Time-stable performance in parallel queues with non-homogeneous and multi-class workloads," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1322–1335, Jun. 2016.

[45] T. Vasantam, A. Mukhopadhyay, and R. R. Mazumdar, "Mean-field analysis of loss models with mixed-Erlang distributions under power-of-d routing," in *Proc. 29th Int. Teletraffic Congr. (ITC)*, vol. 1, Sep. 2017, pp. 250–258.

[46] A. Gorbunova, I. Zaryadov, S. Matyushenko, and E. Sopin, "The estimation of probability characteristics of cloud computing systems with splitting of requests," in *Proc. Int. Conf. Distrib. Comput. Commun. Netw.* Cham, Switzerland: Springer, 2016, pp. 418–429.

[47] F. Poloczek and F. Ciucu, "Contrasting effects of replication in parallel systems: From overload to underload and back," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 44, no. 1, pp. 375–376, Jun. 2016.

[48] A. Thomasian, "Analysis of fork/join and related queueing systems," *ACM Comput. Surv.*, vol. 47, no. 2, p. 17, 2015.

[49] C. Banerjee, A. Kundu, A. Agarwal, P. Singh, S. Bhattacharya, and R. Dattagupta, "Priority based K-Erlang distribution method in cloud computing," *Int. J. Recent Trends Eng. Technol.*, vol. 10, no. 1, p. 135, 2014.

[50] G. Liang and U. C. Kozat, "FAST CLOUD: Pushing the envelope on delay performance of cloud storage with coding," *IEEE/ACM Trans. Netw.*, vol. 22, no. 6, pp. 2012–2025, Dec. 2014.

[51] R. Bitar, P. Parag, and S. El Rouayheb, "Minimizing latency for secure coded computing using secret sharing via staircase codes," *IEEE Trans. Commun.*, vol. 68, no. 8, pp. 4609–4619, Aug. 2020.

[52] M. F. Aktas, E. Najm, and E. Soljanin, "Simplex queues for hot-data download," in *Proc. ACM SIGMETRICS/Int. Conf. Meas. Modeling Comput. Syst.*, Jun. 2017, pp. 35–36.

[53] A. Behrouzi-Far and E. Soljanin, "Redundancy scheduling in systems with bi-modal job service time distributions," in *Proc. 57th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2019, pp. 9–16.

[54] G. Joshi, E. Soljanin, and G. Wornell, "Efficient replication of queued tasks for latency reduction in cloud systems," in *Proc. 53rd Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2015, pp. 107–114.

[55] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2017, pp. 3368–3376.

[56] C. Huang *et al.*, "Erasure coding in windows azure storage," in *Proc. USENIX Annu. Tech. Conf. (USENIX ATC)*, 2012, pp. 15–26.

[57] (2018). *Kubernetes Documentation*. Accessed: Oct. 26, 2018. [Online]. Available: https://kubernetes.io/docs/reference/

[58] B. Hindman *et al.*, "Mesos: A platform for fine-grained resource sharing in the data center," in *Proc. NSDI*, vol. 11, 2011, p. 22.

[59] Q. Ho *et al.*, "More effective distributed ML via a stale synchronous parallel parameter server," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 1223–1231.

[60] W. Dai, A. Kumar, J. Wei, Q. Ho, G. A. Gibson, and E. P. Xing, "High-performance distributed ML at scale through parameter server consistency models," in *Proc. AAAI*, 2015, pp. 79–87.

[61] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous distributed systems," 2016, *arXiv:1603.04467*.

[62] J. Chen, X. Pan, R. Monga, S. Bengio, and R. Jozefowicz, "Revisiting distributed synchronous SGD," 2016, *arXiv:1604.00981*.

[63] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis, "Large-scale matrix factorization with distributed stochastic gradient descent," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2011, pp. 69–77.

[64] J. Dean *et al.*, "Large scale distributed deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1223–1231.

[65] A. Krizhevsky, "One weird trick for parallelizing convolutional neural networks," 2014, *arXiv:1404.5997*.

[66] S. Zhang, A. E. Choromanska, and Y. LeCun, "Deep learning with elastic averaging SGD," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 685–693.

[67] G. S. Grimmett *et al.*, *Probability and Random Processes*. London, U.K.: Oxford Univ. Press, 2020.

[68] B. C. Arnold, *Pareto Distribution*. Hoboken, NJ, USA: Wiley, 2015.

[69] A. Rényi, "On the theory of order statistics," *Acta Math. Hungarica*, vol. 4, nos. 3–4, pp. 191–231, 1953.

[70] B. C. Arnold, N. Balakrishnan, and H. N. Nagaraja, *A First Course in Order Statistics* (Classics in Applied Mathematics). Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2008.

[71] H. A. David, *Order Statistics*. Berlin, Germany: Springer, 2011, pp. 1039–1040.

[72] S. S. Gupta, "Order statistics from the gamma distribution," *Technometrics*, vol. 2, no. 2, pp. 243–262, 1960.

[73] W. Gautschi, "Some elementary inequalities relating to the gamma and incomplete gamma function," *J. Math. Phys.*, vol. 38, nos. 1–4, pp. 77–81, 1959.

[74] M. S. Klamkin and D. J. Newman, "Extensions of the birthday surprise," *J. Combinat. Theory*, vol. 3, no. 3, pp. 279–282, 1967.

Pei Peng received the B.S. degree in information engineering from the South China University of Technology, Guangzhou, China, in 2011, and the M.S. degree in electronics and communication engineering from the Shanghai Institute of Micro-System and Information Technology, Chinese Academy of Sciences, Shanghai, China, in 2014. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Rutgers, The State University of New Jersey, USA. During his Ph.D. studies, he has served as a research assistant and a teaching assistant. His research interests are coding and allocation in distributed computing systems, covert communications, and machine learning. He has received the 2020 ECE Department Teaching Award.

Emina Soljanin (Fellow, IEEE) is currently a Professor at Rutgers University. She is also an Outstanding Alumnus of the Texas A&M School of Engineering. Before moving to Rutgers in 2016, she was a (Distinguished) Member of Technical Staff for 21 years in mathematical sciences research at Bell Labs. Her interests and expertise are wide. Over the past quarter of the century, she has participated in numerous research and business projects as diverse as power system optimization, magnetic recording, color space quantization, hybrid ARQ, network coding, data and network security, distributed systems performance analysis, and quantum information theory. She is the 2019 President of the IEEE Information Theory Society. She served as an Associate Editor for Coding Techniques for the IEEE TRANSACTIONS ON INFORMATION THEORY on the Information Theory Society Board of Governors, and in various roles on other journal editorial boards and conference program committees. She is the 2011 Padovani Lecturer and a 2016/17 Distinguished Lecturer.

Philip Whiting received the B.A. degree from the University of Oxford, the M.Sc. degree from the University of London, and the Ph.D. degree in queueing theory from the University of Strathclyde. After holding a post-doctoral position with the University of Cambridge, his research interests centered on wireless. In 1993, he participated in the Telstra trial of Qualcomm CDMA in South Eastern Australia. He then joined the Mobile Research Centre, University of South Australia, Adelaide. He was a Researcher at Bell Labs from January 1997 to June 2013. He has over 25 patents in applications for DSL vectoring, wireless networks, and location and tracking. He has held visiting positions including ones at Brown University (Maths), University of Korea (engineering), and Vrij University (Maths). For the past three years, he has been a STAR Visiting Scholar with the Maths Department, Eindhoven University of Technology, where his collaborative work includes investigations of both CSMA networks and load balancing in queueing networks. He is currently a Research Professor at Macquarie University, Sydney, Australia, and a Consultant at Telstra for the past two years. Apart from papers in various aspects of telecommunications, he has been an author on various aspects of probability theory, including random Vandermonde matrices, large deviations theory for occupancy models, and more recently random CSMA networks and load balancing in queueing networks. His current research interests include storage systems and wireless mmwave networks. He has received several awards for his work in DSL vectoring and in wireless scheduling.