Efficient Replication for Straggler Mitigation in Distributed Computing

Amir Behrouzi-Far and Emina Soljanin
Department of Electrical and Computer Engineering, Rutgers University
{amir.behrouzifar, emina.soljanin}@rutgers.edu

Abstract—The potential of distributed computing to improve the performance of big data processing engines is contingent on mitigation of several challenges. In particular, by relying on multiple commodity servers, the performance of a distributed computing engine is dictated by the slowest servers, known as stragglers. Redundancy could mitigate stragglers by reducing the dependence of the computing engine on every server. Nevertheless, redundancy could yet be a burden to the system and aggravate stragglers. In this paper, we consider task replication as the redundancy technique and study the optimum redundancy planning to improve the performance of a master-worker distributed computing system. We start with optimum assignment policy of a given set of redundant task to a set of workers. Using the results from majorization theory, we show that if the service time of workers is a stochastically (decreasing and) convex random variable, a balanced assignment of non-overlapping batches of tasks minimizes the average job compute time. With that results, we then study the optimum level of redundancy from the perspective of average job compute time and compute time predictability. We derive the efficient redundancy level as a function of tasks’ service time distribution. We observe that, the redundancy level that minimizes average compute time is not necessarily the same as the redundancy level that maximized compute time predictability. Finally, by running experiments on Google cluster traces, we show that a careful planning of redundancy according to the tasks’ service time distribution can speed up the computing job by an order of magnitude.

Index Terms—redundancy, replication, distributed systems, distributed computing, latency, coefficient of variations.

I. INTRODUCTION

Distributed computing plays an essential role in modern data analytics and machine learning systems [5]–[8]. By enabling parallel task execution, distributed computations can bring considerable speedups to many practical applications, e.g., matrix multiplication [9]–[12], model training in machine learning [13]–[16] and convex optimization [17]–[20]. However, distributed computing algorithms are prone to failures and slow downs more often than their centralized counterparts. In particular, in a synchronous master-worker architecture, where the master waits for every worker to deliver its results before moving to the next iteration, the speed of the computation is dictated by the slowest workers, known as “stragglers” [21]–[23]. With more workers parallelism could be increased, which in turn reduces the workload on workers. Thus, workers should be able to deliver their results faster. However, this is not an indication to a faster computations, because with higher number of workers the probability of failure or slowdown increases. Therefore, the effect of parallelism on the overall speed of the computation is not clear.

Reliability in a distributed system can be improved using redundancy [24]–[26]. Roughly explained, by waiting for a subset of workers, the master node could wrap up the computations without necessarily waiting for the stragglers. Given a fixed budget of $N$ workers, the main question (in task scheduling) is that how many workers should be used as back-ups. In general, as the probability of workers’ slowdown increases the number of back-up servers should grow as well for maintaining reliability. But the exact connection between the slowdown probability of workers and the number or redundant workers is not clear. Several research works are devoted to the modeling of slowdown probability and the efficient redundancy scheduling [1], [2], [23], [26]–[31].

Redundancy in different forms, ranging from simple replication to more complex erasure coding techniques, have been proposed to speed up distributed computations. Coded computing has become a popular research area in recent years [32]–[40]. As an application in machine learning, gradient coding was proposed in [41] as a groundwork for applying erasure coding techniques to distributed gradient descent. Many research works emerged around gradient coding technique. A branch of works proposed efficient erasure coding techniques, e.g. Reed-Solomon [42], LDPC [43], LDGM codes [44] and diagonal codes [45]. Another branch have studied the approximate versions of the gradient descent problem [42], [46]–[48]. Stochastic gradient descent has been studied with redundancy techniques in several papers as well [49]–[52].

Several straggler mitigation techniques have been proposed in the literature. Task replication, as the most essential technique, has been considered in [53]–[56]. Erasure coding techniques, on the other hand, have been shown to have better cost-latency performance under certain scenarios [23], [28]. Generally speaking, replication is simpler to implement since it does not require decoding like erasure codes do. Nevertheless, the higher efficiency of codes compared to replication makes it an appealing option for designing redundancy in distributed computing. In addition, several techniques have been proposed to reduce the pain of the stragglers. Among all are, delayed relaunch of the straggling tasks [29], utilizing non-persistent stragglers [57], and posing a deadline on the response time of.
the workers [58].

Although using erasure codes have been shown to be promising for straggler mitigation and compute time reduction, the contribution of the decoding time in the overall compute time is almost always ignored. The particular problem is that, most decoding algorithms are computationally expensive, since they involve matrix inversion which has $O(N^3)$ time complexity. Therefore, despite the higher efficiency of using erasure coding for redundancy, task replication has been the only technique employed in practice to improve the reliability of computing systems [59]–[64]. Nevertheless, replication is expensive to implement and that for achieving robustness only against single point of failure twice computations are required, which could double the resource usage [65], [66]. Moreover, in the presence of replication updating the tasks and maintaining consistency among replicas could be expensive [67]. These challenges become more salient as the replication (or equivalently redundancy) level increases. On the other hand, a (parallelizable) compute job can be partitioned into pieces and be distributed among the available workers. Partitioned jobs do not update the consistency challenges in the severity that the replicated compute jobs do. They also require less resources. However, with partitions, a compute job is complete only when all the partitions finish their tasks. Thus, a single point of failure could make the entire job unavailable. Therefore, partitioned jobs are prone to failure more than replicated jobs [68].

Now consider a given budget of $N$ workers and a compute job that is $N$-parallelizable, i.e., the job could be split into $N$ parallel tasks. In two extreme cases, this job can be either fully replicated on or fully partitioned into the $N$ workers. We call the former full diversity and the latter full parallelism. Note that, there are possible scenarios between full diversity and full parallelism. For instance, when $N$ is even, it is also possible to partition the tasks into 2 groups and replicate each group on $N/2$ workers. Or group them into $N/2$ groups and replicate each group in 2 places. Thus, depending on the divisibility of $N$ multiple scenarios could be possible. Each possibility lies in one point in diversity-parallelism spectrum, where redundancy grows as moving towards the diversity end of the spectrum.

Optimizing redundancy level for performance improvement was studied in [55], [56] for $(n, k)$ fork-join queuing model and in [69], [70] with coded redundancy. The significance of this problem is also shown in [71] with a new analytical model for servers’ slow-down. In this paper, we study the optimum level of replicated redundancy in a master-worker distributed computing setup from compute time perspective. We characterize the performance of our computing model as a function of the operating point in the diversity-parallelism spectrum. The main contributions of this paper are as follows.

- Given a budget of $N$ workers, an $N$-parallelizable job and a pre-defined level of redundancy the optimum task assignment policy that minimizes the average job compute time is studied. With a pre-defined level of redundancy each worker is assigned with a batch of tasks. Using the results from majorization theory, we show that if the Complementary Cumulative Density Function (CCDF) of the batches’ service time distribution is a stochastically (decreasing) convex function of the batch size the minimum average job compute time could be achieved by balanced assignment of non-overlapping batches. This is an important observation, since the proposed redundancy techniques in the literature either propose overlapping [41] or non-balanced [72] schemes.
- Next, by knowing the optimum batch assignment policy we study the optimum level of redundancy for different tasks’ service time distributions, with each having stochastic convexity property. We observe that, as a general rule of thumb, the higher the randomness in the tasks’ service time the higher the optimum level of redundancy. Furthermore, the jobs with heavy-tail tasks’ service time benefit more from redundancy compared to their exponential-tail counterparts.
- In addition to the average, we study the Coefficient of Variations (CoV) of job compute time. CoV is an important performance metric in the sense that it measures the degree of predictability of job compute time. It is among the most important performance metrics in practice [13] and [73], and has been studied in several research works [74]–[79]. We observe that the optimum level of redundancy that minimizes the coefficient of variations of job compute time is not the same as the level that minimizes the average job compute time. In fact, for exponential distribution, the two optimum points are at the opposite ends of the diversity-parallelism spectrum. This is an important result in that it indicates having a predictable performance may require a job to have longer compute time. This result sheds light on the required cost for performance guarantees in distributed computing systems.
- Finally, we run experiments on the Google cluster traces, using the runtime information of several jobs in the dataset. We collect this information by deep exploration in the events’ time stamps and event types. We observe that tasks with both heavy-tail and exponential-tail behaviour are present in Google clusters. We show that by introducing redundancy the jobs in Google clusters could gain speed up. Furthermore, the optimum level of redundancy depends on the tasks’ service time distribution. As it is also shown by our analysis, jobs with heavy-tail tasks’ service time distribution benefit more from redundancy.

The organization of this paper is as follows. In section [II] we explain the system architecture, compute job model, task replication policies and service time models. The task replication policies, non-overlapping and overlapping batches, are described in section [III]. In Section [IV] the compute time analysis of non-overlapping batches is provided. In section [V] we compare the performance of overlapping batches and non-overlapping batches. Optimum redundancy level for performance improvement is studied in [VI]. We present our experimental results and conclusions in Section [VII] and Section [VIII] respectively.
II. SYSTEM MODEL

A. Distributed Computing Architecture

We study a master-worker distributed computing architecture, as shown in Fig. 1. We will refer to this system as system 1. A compute job in our model is $N$-parallelizable, meaning that it can be split into $N$ tasks and be executed in $N$ workers concurrently. The master node in our model has the following functionalities: 1) forming batches of tasks (task batching), 2) assigning batches to workers (batch assignment), 3) aggregating computation results from the workers (local result aggregation), and 4) generating the overall result (final result generating). For a given compute job and a fixed number of workers, the system 1 is a function of the following factors: 1) the task-to-worker assignment policy, 2) the service time distribution of the workers and, 3) the way workers communicate their computations’ results to the master node. We will discuss each factor in the following subsections.

B. Compute Job Model

In this work, we consider $N$-parallelizable compute jobs that can be divided into $N$ smaller tasks to be executed concurrently. We assume tasks have no dependencies. Each task is redundantly assigned to a subset of workers to improve the computation reliability. To reduce the communication overhead, each worker sends the computations result to the master once it finished executing all of its assigned tasks. After receiving the local results from a large enough number of workers, the master node generates the overall result of the compute job. This model can be applied to different problems, e.g., matrix multiplication [9], [12], [34], gradient based optimizers [17], [41], model training in machine learning problems [41], [42]. For instance, consider the problem of optimizing a model $\beta$ with distributed gradient descent algorithm, to reduce a loss function $L(\beta; D)$, over a chunked dataset $D = \{D_1, D_2, D_3, D_4\}$. The model at the $i$th iteration of the gradient descent algorithm is given by

$$\bar{\beta}_i = \bar{\beta}_{i-1} - \gamma \nabla L(\bar{\beta}_{i-1}, D),$$

(1)

where $\gamma$ is the step size. We can write (1) as,

$$\bar{\beta}_i = \bar{\beta}_{i-1} - \gamma \nabla \sum_{k=1}^{4} L(\bar{\beta}_i, D_k).$$

(2)

The summation in the RHS of (2) can be redundantly distributed among four workers. Two examples of replication policies are provided in Fig. 2. In Example 1, the results from the fastest worker among $W_1, W_2$ and the fastest worker among $W_3, W_4$ are enough for the master node to generate the overall compute result. By changing the task-to-worker assignment in Example 2, the result form the fastest group of workers among $W_1, W_2$ and $W_3, W_4$ is sufficient for generating the overall result.

C. Task Replication Policies

To improve computations reliability, each task is assigned to multiple workers. We abstract the redundant distribution of tasks among workers into a two-stage process: 1) grouping tasks (redundantly) into equal-sized batches, and 2) assigning batches (redundantly) to the workers. Note that, batches could be non-overlapping, when the set of tasks is simply chopped into smaller pieces, or they could overlap if each task is available in more than one batch. Nevertheless, the batch sizes will be the same in both cases. Therefore, there would be more batches if they overlap. Assuming that batch size is $N/B$, where $B|N$, the number of batches is an integer in the range of $[B, N]$.

D. Service Time Model

We define $T_{i,j}$ in section IV as the service time of batch $i$ at the $j$th worker it’s assigned to. The service time of a batch is the interval between the time a batch starts service and the time it associated computation result gets delivered to the master node. We assume that $T_{i,j}$s are all independent and identically distributed (i.i.d) random variables. In section VI we define $\tau$ as the i.i.d random variable for the service time of tasks. We study the computing time of system under different service time distributions of batches/tasks, defined as follows.
1) Exponential Distribution: It is a highly referred distribution for life time of tasks in queuing theory literature [80]. If \( T_{i,j} \sim \text{Exp}(\mu) \) then,
\[
\Pr\{T_{i,j} > t\} = 1 - e^{-\mu t}
\]
where \( \mu \) is the service rate and we assume it is identical across the batches/tasks. \( \mathbb{I}(\cdot) \) is the indicator function.

2) Shifted-Exponential Distribution: It is a well studied model and is of special interest when the service time of a task can not be below a minimum value \([26]\). If \( T_{i,j} \sim \text{SExp}(\mu, \Delta) \) then,
\[
\Pr\{T_{i,j} > t\} = 1 - (1 - e^{-\mu(t-\Delta)})^n
\]
where \( \mu \) is the service rate and \( \Delta \) is the shift parameter, defined as the minimum possible service time. We assume \( \mu \) and \( \Delta \) are identical across the batches/tasks.

3) Pareto Distribution: It is a well studied class of heavy tail distributions, which well captures the service behaviour in the practical systems. If \( T_{i,j} \sim \text{Pareto}(\alpha, \sigma) \) then,
\[
\Pr\{T_{i,j} > t\} = 1 - \left( \frac{\sigma}{\sigma + t} \right)^\alpha
\]
where \( \alpha \) and \( \sigma \) are shape and scale parameters, respectively. We assume both parameters are identical across the batches/tasks.

### III. Task Replication Policies

A task replication policy can be viewed as a two-stage process, 1) task batching, and 2) batch assignment. Task batching schemes can be categorized into, a) non-overlapping batches, or b) overlapping batches. In what follows, we will discuss each batching scheme in detail.

#### A. Non-overlapping Batches

Under non-overlapping batching policy, the set of \( N \) tasks is split into \( B \) equal-size batches, giving the batch size \( N/B \). When \( B < N \), batches can be assigned redundantly to the \( N \) workers. Since the intersection of the batches are empty, the batch at a given worker either completely overlaps or do not overlap at all with the batches at any other worker. Therefore, a batch of tasks is considered to be computed once one worker (among the ones hosting the batch) delivers its local result.

The random batch assignment in which each worker draws a batch, uniformly at random with replacement from the pool of batches was studied in [82], and can be naturally modeled as the Coupon Collection (CC) problem. It was shown in the same work that the random assignment reduces the compute time compared to a deterministic assignment in [41]. We will show in the following section that the imbalance task replication among workers, which resulting from the random assignment, adversely affects the computation time. On the other hand, since some of the batches will be drawn only once, there will be no failure tolerance for a worker if its tasks are not replicated at any other worker. Furthermore, by random assignment, there is always a non-zero probability that some batches do not get selected at all, leading to an inaccurate computations result. We analyze this probability in the following.

Let \( n \) be the random variable for the number of workers required for covering all the batches in random assignment policy. The following proposition provides the batch coverage probability with the random batch-to-worker assignment.

**Lemma 1.** The probability of covering \( B \) batches with \( N \) workers, with random batch-to-worker assignment is given by,
\[
\Pr\{n \leq N\} = \frac{B!}{B^N} \binom{N}{B},
\]
where \( \binom{N}{B} \) is the Stirling number of second kind [81], given by,
\[
\binom{N}{B} = \frac{1}{B!} \sum_{k=0}^{B} (-1)^{B-k} \binom{B}{k} k^N.
\]

**Proof.** The probability of covering \( B \) batches with exactly \( N \) workers is given in [82], as
\[
\Pr\{n = N\} = \frac{B!}{B^N} \binom{N-1}{B-1}.
\]
Therefore,
\[
\Pr\{n \leq N\} = \frac{B!}{B^N} \sum_{n=0}^{B} \binom{N-1}{n-1} \binom{n}{B-1}
= \frac{B!}{B^N} \sum_{n=0}^{N-1} \left( \frac{1}{B^{n+1}} \binom{n}{B-1} \right)
= \frac{B!}{B^N} \sum_{n=0}^{N-1} \binom{N-1}{n-1} \binom{n}{B-1}
= \frac{B!}{B^N} \binom{N-1}{B}.
\]
The probability of covering all batches, \([6]\), versus the number of batches is plotted for several values of \(N\) in Fig. [3]. In order to cover all \(B\) batches with high probability the number of workers should be much larger than the number of batches. For instance, with \(N = 100\) only \(B = 10\) batches can be covered with high probability. Therefore, randomly assigning batches to workers is not a good practice since the probability of being able to pick each batch at least once reduces very fast as \(B\) increase. That could result in an inaccurate computing result derived from only a subset of tasks.

### B. Overlapping Batches

With overlapping batches, tasks are redundantly grouped in \(N\) overlapping batches, each assigned to a worker node. To be able to compare the computing time, we keep the same batch size in both overlapping and non-overlapping policies. Thus, the batch size with overlapping batches is \(N/B\), where \(B\) is the number of batches in non-overlapping scheme. Note that with overlapping batches, the number of batches is equal to the number of workers and, therefore, we need not decide about the number of workers assigned to each batch. Besides, the challenge in designing overlapping batches is to decide which batches and how much should they overlap for faster computations. This question in general is very complex to answer, since the size of the problem increases exponentially with the number of batches/workers. Hence, we will study it under some reasonable assumptions. In particular, we assume that the set overlapping batches could be divided into subsets, such that each task appears exactly once in each subset. In other words, each group hosts all the tasks, divided into overlapping batches of size \(N/B\). This is a reasonable assumption in the sense that it enforces fairness in task-to-worker assignment. With this policy, the question is how we should arrange the tasks in each subset to have faster computations, which will be answered in section [V].

### IV. Compute Time with Non-Overlapping Batches

In this section we analyze the compute time in system \(\square\) with non-overlapping batches, under the service time distributions discussed in section [III].

Let \(N_i\) be the number of workers hosting batch \(i\), and define \(\bar{N} = (N_1, N_2, \ldots, N_B)\) as the batch assignment vector. As illustrated in Fig. [4], \(N_i\) workers start computations over batch \(i\) simultaneously. Thus, the result from the fastest worker, out of \(N_i\), is sufficient for the master to recover the computations over batch \(i\). Accordingly, the \(i\)'th batch compute time is the first order statistics of \(N_i\) i.i.d random variables, given by

\[
T_i = \min(T_{1,i}, T_{2,i}, \ldots, T_{N_i,i}), \quad \forall i \in \{1, 2, \ldots, B\}. \tag{8}
\]

For generating the overall result, the master node has to wait for the results of computations over all batches. In other words, the overall result can be generated only after the slowest group of workers (hosting the same batch) deliver the local result. Hence, the job compute \(T\) could be written as,

\[
T = \max(T_1, T_2, \ldots, T_B). \tag{9}
\]

The question we address next is how should one redundantly assign \(B\) non-overlapping batches of task among \(N > B\) workers to achieve the minimum average compute time in system \(\square\). To that end, we define the following concepts.

**Definition 1.** The real valued random variable \(X\) is greater than or equal to the real valued random variable \(Y\) in the sense of usual stochastic ordering, shown by \(X \geq_{st} Y\), if their CCDF satisfy

\[
\Pr\{X > \beta\} \geq \Pr\{Y > \beta\}, \quad \forall \beta \in \mathbb{R}, \tag{10}
\]

or equivalently,

\[
\mathbb{E}[\phi(X)] \geq \mathbb{E}[\phi(Y)],
\]

for any non-decreasing function \(\phi\).

**Definition 2.** The random variable \(X(\theta)\) is stochastically decreasing and convex if its CCDF is a decreasing and convex function of \(\theta\).

**Definition 3.** For any \(V_p = (v_{p1}, v_{p2}, \ldots, v_{pM})\) in \(\mathbb{R}^M\), the rearranged coordinate vector \(V[p]\) is defined as \(V[p] = (v_{[p1]}, v_{[p2]}, \ldots, v_{[pM]})\), the elements of which are the elements of \(V\) rearranged in decreasing order, i.e., \(v_{[p1]} > v_{[p2]} > \cdots > v_{[pM]}\).

**Definition 4.** Let \(V[p] = (v_{[p1]}, v_{[p2]}, \ldots, v_{[pM]})\) and \(V[q] = (v_{[q1]}, v_{[q2]}, \ldots, v_{[qM]})\) be two rearranged coordinate vectors in \(\mathbb{R}^M\). Then \(V_p\ majorizes\ \ V_q\), denoted by \(V_p \succeq V_q\), if

\[
\sum_{i=1}^m v_{[pi]} \geq \sum_{i=1}^m v_{[qi]}, \quad \forall m \in \{1, 2, \ldots, M\},
\]

and

\[
\sum_{i=1}^M v_{[pi]} = \sum_{i=1}^M v_{[qi]}.
\]
Definition 5. A real valued function \( \phi : \mathbb{R}^m \to \mathbb{R} \) is schur convex if for every \( V \) and \( W \) in \( \mathbb{R}^m \), \( V \geq W \) implies \( \phi(V) \geq \phi(W) \).

Definition 6. A real valued random variable \( Z(\bar{x}) \), \( \bar{x} \in \mathbb{R}^m \), is stochastically schur convex, in the sense of usual stochastic ordering, if for any \( \bar{x} \) and \( \bar{y} \) in \( \mathbb{R}^m \), \( \bar{x} \succeq \bar{y} \) implies \( Z(\bar{x}) \geq Z(\bar{y}) \).

Lemma 2. If the batch assignment \( \bar{N}_1 = (N_{11}, \ldots, N_{1B}) \) majorizes the batch assignment \( \bar{N}_2 = (N_{21}, \ldots, N_{2B}) \), that is, \( \bar{N}_1 \succeq \bar{N}_2 \), then the corresponding completion times \( T(\bar{N}_1) \) and \( T(\bar{N}_2) \) satisfy

\[
\mathbb{E}[T(\bar{N}_1)] \geq \mathbb{E}[T(\bar{N}_2)],
\]

if the service time of the batches are i.i.d stochastically decreasing and convex random variables.

Proof. The job compute time for the batch assignment policy \( \bar{N}_k \forall k \in \{1, 2\} \), is given by

\[
T(\bar{N}_k) = \max(T_{k1}, T_{k2}, \ldots, T_{kB}),
\]

where \( T_{ki} i \in \{1, 2, \ldots, B\} \) is stochastically decreasing and convex random variable. Since max(.) is a schur convex function \( \phi \), \( T(\bar{N}_k) \) is stochastically decreasing and schur convex function of \( \bar{N}_k \). Hence, by Definition 6 \( \bar{N}_1 \succeq \bar{N}_2 \) implies \( T(\bar{N}_1) \geq T(\bar{N}_2) \) in the sense of usual stochastic ordering, which in turn implies that for any non-decreasing function \( \phi \),

\[
\mathbb{E}[\phi(T(\bar{N}_1))] \geq \mathbb{E}[\phi(T(\bar{N}_2))].
\]

Substituting \( \phi \) by the unit ramp function completes the proof.

Lemma 3. The balanced batch assignment, defined as \( \bar{N}_b = (N/B, N/B, \ldots, N/B) \) with \( N \) and \( B \) being the respective number of workers and batches, is majorized by any other batch assignment policy.

Proof. See [84].

Next we find the optimum batch assignment policy with different batch service time distribution.

A. Exponential Distribution

With exponential service time distribution of batches \( T_{ij} \sim \text{Exp}(\mu) \), the compute time of batch \( i \in \{1, 2, \ldots, B\} \) is the minimum of \( N_i \) i.i.d exponential random variables. Hence,

\[
T_i \sim \text{Exp}(N_i \mu), \quad \forall i \in \{1, 2, \ldots, B\}. \tag{11}
\]

Therefore, \( T \) is the maximum of \( B \) independent exponential random variables. Thus, the following follows from Lemma 2.

Theorem 1. With exponential service time distribution of batches \( T_{ij} \sim \text{Exp}(\mu) \), among all (non-overlapping) batch assignment policies, the balanced assignment achieves the minimum expected job compute time.

B. Shifted-Exponential Distribution

With shifted-exponential model, the service time of a batch consists of a deterministic part and a random part. The deterministic part can be associated with the minimum required time, imposed by physical constraints, for a job to be executed in a system. This part may differ depending on the system. On the other hand, the random part can be associated with the slow-down a job experiences during the execution, and could vary across the jobs. The following corollary gives the optimum batch assignment policy when the service time of batches has shifted-exponential distribution.

Corollary 1. With shifted-exponential service time distribution of batches \( T_{ij} \sim \text{SExp}(\Delta, \mu) \), the minimum expected job compute time is achieved by the balanced assignment of non-overlapping batches.

C. Pareto Distribution

With pareto model, the service time of a batch consists of a deterministic part and random, heavy tail-part. Similar to shifted-exponential mode, the deterministic part can be associated with the minimum required service time imposed by a system. However, the random part, unlike the shifted-exponential model, has the heavy-tail property. Meaning that, the tail of the distribution decays slower than exponential and, thus, the probability of having very long service time is higher than that with shifted-exponential service time distribution. The following lemma gives the optimum batch assignment when the service time of batches has pareto distribution.

Theorem 2. With pareto service time distribution of batches \( T_{ij} \sim \text{Pareto}(\sigma, \alpha) \), the minimum expected job compute time is achieved by balanced assignment of non-overlapping batches.

We have showed so far that if the service time of batches is a convex random variable, or a shifted version of a convex random variable (e.g. shifted-exponential and pareto), and batches do not overlap, then the balanced assignment achieves the minimum average job compute time. The case of concave random variables, e.g. weibull and gamma with shape parameters \( \alpha > 1 \), is left as an open problem for future studies. In the following section, we compare the average job service time with overlapping batches with that of non-overlapping batches.

V. Compute Time with Overlapping Batches

Recall the original problem of assigning a task set of size \( N \) redundantly among \( N \) workers. Each worker is assigned with \( N/B \) tasks, for a given parameter \( B|N \). There are many possible ways to group the \( N \) tasks into \( N \) batches. Here we focus on the schemes where the set of overlapping batches could be divided into subsets, such that each task appears exactly once in each subset. The reason for this choice is the fairness in task-to-worker assignment; each task is hosted by equal number of workers. Therefore, each subset comprises of one copy all the tasks. Note that, the non-overlapping policy is an especial case of these schemes, where the inside the subsets a batch does not overlap with any other batch.
As another especial case, consider the scheme where the task set is divided into $N$ overlapping batches, each with size $N/B$, in a cyclic order as follows. The first batch consists of tasks 1 through $N/B$, the second batch consists of tasks 2 through $N/B + 1$, and so on. This is scheme 1 in Fig. 5. Note that, the cyclic scheme and non-overlapping policy are the two end of the same spectrum. With cyclic scheme the number of batches that share at least one task with any given batch is maximized, whereas with non-overlapping policy this number is minimized. Precisely, with cyclic scheme, each batch shares at least one common task with 2 ($N/B$) other batches. With non-overlapping batches this number is $N/B - 1$. With any other batching scheme this number is greater than $N/B - 1$ and smaller than 2 ($N/B - 1$), an example of which is given as scheme 2 in Fig. 5. In what follows, we will provide analytic comparison of the job compute time of system 1 with three different batching schemes, provided in Fig. 5. To that end, we analyze the especial case $N = 6$ and $B = 3$. The extension of our method to general $N$ is straightforward.

Consider system 1 with $N = 6$, $B = 3$, and three different batching schemes, shown in Fig. 5. In each scheme, there are two subsets of batches that consist of a single copy of every task. The subsets are shown by different shapes. Let $X_i$ for $i = 1, 2, \ldots, 6$ be the i.i.d random variable of batch $i$ compute time. Let us assume, without loss of generality, that $W_1$ is the fastest worker, delivering its local result before the rest of the workers. Then the job compute time of scheme 1 is

$$T^{(1)} = \min (\max (X_3, X_5), \max (X_2, X_4, X_6)).$$

With scheme 2, the job compute time could be written as,

$$T^{(2)} = \min (\max (X_3, \min (X_5, X_6)),\max (\min (X_2, X_4), \min (X_5, X_6))).$$

Comparing (12) and (14), it can be verified that $E[T^{(2)}] < E[T^{(1)}]$, since

$$E[\max (X_3, \min (X_5, X_6))] < E[\max (X_3, X_5)],$$

and

$$E[\max (\min (X_5, X_6), \max (X_2, X_4))] < E[\max (X_2, X_4, X_6)].$$

On the other hand, for scheme 3,

$$T^{(3)} = \max (\min (X_3, X_4), \min (X_5, X_6)).$$

In order to be able to compare the job compute time with scheme 3, we rewrite (14) as follows:

$$T^{(2)} = \max (\min (X_3, \max (X_2, X_4)), \min (X_5, X_6)).$$

This essentially means that, the balanced assignment of non-overlapping batches achieves the minimum expected computing time when compared to overlapping batch assignment. This result is also approved by our simulation results in Fig. 6.

The first argument of the utmost max functions in (15) and (16) are compared as,

$$E[\min (X_3, X_4)] < E[\min (X_3, \max (X_2, X_4))].$$

Therefore, $E[T^{(3)}] < E[T^{(2)}]$. Accordingly, the expected completion times of three batching policies in Fig. 5 are compared as follows:

$$E[T^{(3)}] < E[T^{(2)}] < E[T^{(1)}].$$

VI. EFFECT OF BATCH SIZE ON THE COMPUTING TIME

Replication increases availability in distributed systems [86]–[88]. With higher availability, the possibility of loosing a compute job because of the faulty workers reduces. Therefore, computations with redundancy are more reliable. Furthermore, by not depending on a single copy of a job, a user may experience a shorter job completion time by waiting for the fastest copy. In short, replication increases the diversity in job compute time. On the other hand, replication increases the storage and computing costs, makes updates more complex.
and brings inconsistency issues to the system. Partitioning, on the other hand, improves performance and resource utilization by enabling concurrent executions of smaller tasks, comprising the compute job \[89, 90\]. In short, it increases the \textit{parallelism} in the job execution. However, non-balanced partitions, resulting in skewed latency \[20\], and reliance on every partition, which makes an execution as fast as the slowest copy \[26\], are among the issues that make partitioning a challenge in distributed systems.

In our model given in Fig. 1 replication and partitioning are the two end of diversity-parallelism spectrum. In the high diversity end, redundancy is maximum and the compute job is replicated on every worker. In high parallelism end, there is no redundancy and each task exists only on one worker. In this section we look at the long-standing problem of efficient replication/partitioning design in a distributed system from the performance perspective. In particular, with average job compute time as the performance metric we study the optimum point in the diversity-parallelism spectrum. We also study the coefficient of variations of the job compute times as a performance metric, which is defined as the ratio between the standard deviation and the expected value of a random variable. It measures the variation in the completion time with respect to its average and is a metric to measure the predictability of the job compute time \[74-79\]. Predictability is among the major concerns in distributed systems since it enables performance guarantees \[21\].

We have so far shown that, with a given level of redundancy, the balanced replication of non-overlapping batches minimizes the average job compute time in system \[1\]. With that in mind, we next study the effect of redundancy level on the job compute time. Note that, in our model the level of redundancy is captured by the parameter \(B\) such that with smaller \(B\) redundancy level is higher. In other words, we try to understand that which one is more beneficial; having a few large batches and replicating them in many workers, i.e. high redundancy and high diversity, or having many small batches and replicating them in fewer workers, i.e. low redundancy and high parallelism. In both cases we distribute batches in the balanced fashion.

To understand the effect of redundancy level on the job compute time, a size-dependent batch service time model is required. We use the model proposed by \[71\], where the service time of a batch is the random variable formed by the product of the batch size, i.e. number of tasks in the batch, and the random variable associated with the service time of a task \(\tau\). Therefore, the service time of a batch with size \(N/B\) is \(N\tau/B\). In the rest of the paper we assume \(N\) is an even number greater than 4, unless otherwise is stated. The extension of the results to the odd numbers is straightforward. Next, we will analyze the effect of redundancy level on the job compute time for different task service time distribution.

\textbf{A. Exponential Distribution}

The following theorem gives the optimum operating point in diversity-parallelism spectrum, form average job compute time perspective, when the service time of tasks follows exponential distribution.

\textbf{Theorem 3.} With exponential service time distribution of tasks \(\tau \sim \text{Exp}(\mu)\), maximum diversity minimizes the average job compute time.

With exponential distribution, the service time of a job is random in \((0, \infty)\). In other words, a job can take arbitrarily short or arbitrarily long time, although with negligible probability, to be completed. Accordingly, high diversity increases the chance of having shorter compute time in some workers. Therefore, maximum diversity minimizes the average job compute time.

Let’s define \(H_{(B,1)} = \sum_{k=1}^{B} 1/k\) and \(H_{(B,2)} = \sum_{k=1}^{B} 1/k^2\) as the first order and second order harmonic numbers, respectively. The following theorem characterizes the coefficient of variations of job compute time.

\textbf{Lemma 4.} With exponential service time distribution of tasks \(\tau \sim \text{Exp}(\mu)\), the coefficient of variations of the job compute time is given by,

\[
\text{CoV}[T] = \frac{\sqrt{H_{(B,2)}}}{H_{(B,1)}} \quad (18)
\]

\textbf{Theorem 4.} With exponential service time distribution of tasks \(\tau \sim \text{Exp}(\mu)\), the coefficient of variations of job compute time is minimized at full parallelism.

From the average job compute time perspective maximum diversity and from the coefficient of variations perspective maximum parallelism are the optimum operating points for system \[1\]. This result shows that, with \(\tau \sim \text{Exp}(\lambda)\), there exist an inevitable trade-off between the two performance metrics. With full diversity, jobs may experience small average compute time but relatively high fluctuations in the performance. On the other hand, with full parallelism, a job may experience large average compute time with relatively smaller fluctuations. From a system administrator point of view, may neither operating points be interesting, since high fluctuations and large average of compute time could both be undesirable. Accordingly, she may choose a middle point in diversity-parallelism spectrum to operate the system.

\textbf{B. Shifted-Exponential Distribution}

For shifted-exponential service time distribution of tasks, the following theorem gives the optimum level of redundancy that minimizes the average job compute time.

\textbf{Theorem 5.} With shifted-exponential service time distribution of tasks \(\tau \sim S\text{Exp}(\Delta, \mu)\), the optimum \(B\) achieving the minimum average job compute time is the solution of the following discrete unconstrained optimization problem,

\[
\min_{B \in F_{B}} \quad \frac{N\Delta}{B} + \frac{1}{\mu} H_{(B,1)}, \quad (19)
\]

where \(F_{B}\) is the set of all feasible values for \(B\).

Form \[19\] it can be verified that for large values of \(\Delta\) and \(\mu\), the term \(N\Delta/B\) is dominated in the objective function and lower redundancy is more beneficial. On the other hand, for small values of \(\Delta\) and \(\mu\), the term \(1/\mu H_{(B,1)}\) is dominated and higher redundancy is more beneficial. Note that, the
former case corresponds to lower randomness and the latter corresponds to the higher randomness in the service time of tasks. This result is in line with intuition, since the higher the randomness, the higher probability that a worker experiences severe slow-down. Thus, more redundancy is required to alleviate the effect of high randomness.

The following theorem characterizes the connection between the optimum operating point in the diversity-parallelism spectrum and the parameters $\Delta$ and $\mu$.

**Theorem 6.** With shifted-exponential service time distribution of tasks $\tau \sim SE\text{xp}(\Delta, \mu)$, the optimum operating point of system $[2]$ forms the average job compute time perspective, in the diversity-parallelism spectrum is,

- at full diversity when $\Delta \mu < 1/N$,
- at a middle point when $1/N \leq \Delta \mu \leq \sum_{k=N/2+1}^{N} 1/k$, and
- at full parallelism when $\sum_{k=N/2+1}^{N} 1/k < \Delta \mu$.

When the product $\Delta \mu$ lies in the range $(1/N, \sum_{k=N/2+1}^{N} 1/k < \Delta \mu)$, the following corollary simplifies the optimization problem in [19].

**Corollary 2.** With shifted-exponential service time distribution of tasks $\tau \sim SE\text{xp}(\Delta, \mu)$ and $1/N \leq \Delta \mu \leq \sum_{k=N/2+1}^{N} 1/k$, the optimum operating point of system $[2]$ (from the average compute time point of view) in the diversity-parallelism spectrum is the solution of the following discrete unconstrained optimization problem,

$$
\min_{B \in F_B} |B - N\Delta \mu|,
$$

where $F_B$ is the set of all feasible values for $B$.

The value of the optimal $B$, minimizing the average job compute time, increases if any of $N$, $\Delta$ or $\mu$ is increased. This is inline with intuition since the larger the product $\Delta \mu$ the smaller the randomness in the completion time, which makes parallelism more efficient. The average job compute time for shifted-exponential service time distribution of tasks is plotted in Fig. 7 for $N = 100$ and $\Delta = 0.05$. For this set of parameters, $1/N = 0.01$ and $\sum_{k=N/2+1}^{N} 1/k \approx 0.69$. Hence, for $\mu < 0.2$ full diversity should minimize the average job compute time. For $0.2 \leq \mu \leq 13.8$ the optimum point should be in the middle of the spectrum. Finally, for $\mu > 13.8$ full parallelism should minimize the average job compute time. All these regions could be verified in Fig. 7.

**Lemma 5.** With shifted-exponential service time distribution of tasks $\tau \sim SE\text{xp}(\Delta, \mu)$, the coefficient of variations of the job compute time is given by,

$$
CoV[T] = \frac{\sqrt{H(B,2)}}{N\Delta \mu_B + H(B,1)}.
$$

**Theorem 7.** With shifted-exponential service time distribution of tasks $\tau \sim SE\text{xp}(\Delta, \mu)$ and $N > 4$, the minimum coefficient of variations of job compute time in the diversity-parallelism spectrum is,

- at full parallelism when $\Delta \mu < 3/(\sqrt{5} - 1)N$,
- at either end of the spectrum when $3/(\sqrt{5} - 1)N \leq \Delta \mu \leq H(1)\sqrt{H(2)} - H(1)\sqrt{H(2)}$, $\Delta \mu > 2\sqrt{H(2)} - \sqrt{H(2)}$,
- at full diversity when $\Delta \mu \leq H(1)\sqrt{H(2)} - H(1)\sqrt{H(2)}$.

When task service time follows shifted-exponential distribution, the optimum operating point maximizing the job compute time predictability is either full diversity or full parallelism. When randomness in the task service time is high, i.e. $\Delta \mu \in (-\infty, 3/(\sqrt{5} - 1)N)$, then splitting the job into $N$ parallel tasks and running them concurrently maximizes the job compute time predictability. With small $\Delta \mu$ the coefficient of variation with shifted-exponential model (2) reduces to that of exponential model (13), where full parallelism is optimal. When $\Delta \mu$ is large, on the other hand, the term $\frac{N\Delta \mu}{B}$ is dominated in the denominator of (2) and smallest $B$, i.e. full diversity, is optimal. Furthermore, by looking at the three regions given in Theorem 6 it can be seen that for very small or very large values of $\Delta \mu$ the optimum operating points minimizing the average compute time and maximizing compute time predictability lay at the apposite end of the spectrum. In other words, in order to have predictable performance, longer average compute time may be imposed to a job.

For middle values of $\Delta \mu$ and large enough $N$ the following corollary gives the optimum operating point in the diversity-parallelism spectrum.

**Corollary 3.** With Shifted Exponential service time distribution of tasks $\tau \sim SE\text{xp}(\Delta, \mu)$, $N > 11$ and $3/(\sqrt{5} - 1)N \leq \Delta \mu \leq H(1)\sqrt{H(2)} - H(1)\sqrt{H(2)}$, the minimum coefficient of variation in diversity-parallelism spectrum is,

- at full parallelism when $\Delta \mu < H(1)/\sqrt{H(2)} - 1$,
- at full diversity when $H(1)/\sqrt{H(2)} - 1 \leq \Delta \mu$. 

![Fig. 7. Average job compute time with $\tau \sim SE\text{xp}(0.05, \mu)$, versus the number of batches, for different values $\mu$. The minimum value of $\text{EE}[T]$ moves toward the full parallelism point as $\mu$ increases.](image)
Parallelism

The coefficient of variations of job compute time is plotted in Fig. 8 for $N = 100$ and $\Delta = 0.05$. With those parameters, $H_{(N,1)}/N(\sqrt{H_{n,2}} - 1) \approx 0.04$. Thus, for $\mu < 0.04/\Delta \approx 0.8$ full diversity and for $\mu > 0.8$ full parallelism should be optimal. These results agree with the plots in Fig. 8. Finally, we present the results for the limit case, when $N \to \infty$ in the following corollary.

**Corollary 4.** With shifted-exponential service time distribution of tasks $\tau \sim SExp(\Delta, \mu)$, as $N \to \infty$ the minimum of $\text{CoV}[T]$ occurs at full diversity.

From the presented results for shifted-exponential service time model we can conclude that, the optimum level of redundancy may not be the same from the expected value and the coefficient of variations of job compute time. In fact, for small and large values of $\Delta \mu$ product, the optimum points are at the opposite ends of the spectrum. In other words, the levels of redundancy that minimizes the average compute time results in large coefficient of variations and vice versa. Thus, there is an inevitable trade-off between the average value and the coefficient of variations of job compute time, when service time of the tasks follow shifted-exponential distribution. As a rule of thumb, when $\Delta \mu$ is small, the average job compute time is smaller at high diversity and the coefficient of variations of job compute time is smaller at high parallelism. Whereas, when $\Delta \mu$ is large, the average job compute time is smaller at high parallelism regime and the coefficient of variations is smaller at high diversity.

**C. Pareto Distribution**

For pareto service time distribution of tasks the following theorem gives the optimal operating point in diversity-parallelism spectrum that minimizes the average job compute time.

**Theorem 8.** With pareto service time distribution of tasks $\tau \sim \text{Pareto}(\sigma, \alpha)$, the optimum $B$ achieving the minimum average job compute time is the solution of the following discrete unconstrained optimization problem,

$$\min_{B \in F_B} \frac{N \sigma}{B} \cdot \frac{\Gamma(B + 1) \cdot \Gamma(1 - B/N \alpha)}{\Gamma(B + 1 - B/N \alpha)}, \quad (22)$$

where $\Gamma(\cdot)$ is the Gamma function and $F_B$ is the set of all feasible values for $B$.

With pareto service time model, the optimum level of redundancy only depends on the shape of the distribution $\alpha$. This can be verified from (22), where the scale parameter $\sigma$ appears only as a multiplier. The following theorem further clarifies the solution of the optimization problem (22).

**Theorem 9.** With pareto service time distribution of tasks $\tau \sim \text{Pareto}(\sigma, \alpha)$, the minimum average job compute time in the diversity-parallelism spectrum is

- at a middle point when $1 < \alpha < \alpha^*$, and
A pareto random variable with large $\alpha$ has lighter tail and thus less randomness. Therefore, with large enough $\alpha$ it is intuitively correct that full parallelism should be optimal. With smaller values of $\alpha$, redundancy may be required to reduce the randomness and thus slow-down. Hence, the optimal operating point should move towards the full diversity end of the spectrum. For $N = 100$ and $\sigma = 1$ the average job compute time is plotted in Fig. 9. With those parameters, equation (74) has the solution $\alpha^* \approx 4.7$. Thus, for $\alpha < 4.7$ the optimum $B$ should be in the middle of the diversity-parallelism spectrum and for $\alpha > 4.7$ the optimum $B$ should be at full parallelism, which can be verified by the plots in Fig. 9.

**Lemma 6.** With pareto service time distribution of tasks $\tau \sim \text{pareto}(\sigma, \alpha)$, the coefficient of variations of job compute time is given by,

$$CoV(T) = \sqrt{\frac{\Gamma(B + 1 - B/N\alpha).\Gamma(1 - 2B/N\alpha)}{\Gamma(B + 1 - 2B/N\alpha).\Gamma(1 - B/N\alpha)} - 1.} \quad (24)$$

Similar to the average job compute time, the coefficient of variations with pareto service time model has no dependency with the scale parameter $\sigma$ of the distribution. The following theorem gives the optimum level of redundancy that minimizes the coefficient of variation given in (24).

**Theorem 10.** With pareto service time distribution of tasks $\tau \sim \text{pareto}(\sigma, \alpha)$, the coefficient of variations of job compute time is minimized at full diversity.

Fig. 11 shows the coefficient of variations of job compute time with pareto service time model. The optimum operating point is at full diversity, regardless of the value of $\alpha$. However, full diversity maximizes the average compute time, as it is shown in Fig. 9. This result indicates the trade-off between the average and the coefficient of variations of the compute time, with heavy-tail service time distribution of tasks. In other words, high average compute time is the price to pay for more predictable performance.

**VII. EXPERIMENTAL RESULTS**

To illustrate the effect of redundancy on practical systems, we run experiments using the data from the Google cluster traces [91]. In this dataset, for each job, several attributes are recorded. Each job consists of several tasks, each being scheduled on a worker server. The recorded information for each task includes, among others, its scheduling and finish times. We recorded the service time of a task as the difference between its finish time and scheduling time. By recording the service time of the tasks, we created a data set for each job, representing its tasks service time.

Foremost, we observed that the tasks’ service time could follow both heavy-tail or exponential-tail behaviours, depending on the job. For instance, in Fig. 11 jobs 1 through 4 have exponential decay in tail probability, whereas jobs 5 through 10 have almost linear (heavy-tail) decay.

Fig. 12 shows the average job compute time, normalized by the no-redundancy time, for jobs with exponential decay in the tail probability. As it was concluded from our analysis on shifted-exponential distribution, since the shift parameter of the tasks service time is large (this value is 10 for jobs 1 through 3 and 1000 for job 4), full parallelism is optimal. Nevertheless, job 5, which has a linear decay in the tail probability, has its minimum completion time at $B = 50$.

Fig. 13 shows the normalized average job compute time versus the number of batches, where task service times have the heavy-tail behaviour. The minimum job compute time for all the jobs occur somewhere between full parallelism and full diversity. In other words, neither replicating nor partitioning...
the job are optimal. This observation is inline with our analysis of pareto service time distribution of the tasks. The optimum level of redundancy, however, depends on the type of jobs. For instance, while jobs 6, 8, 9 and 10 have their minimum average completion time at $B = 20$, with job 7 the optimum value is $B = 10$.

VIII. CONCLUSION

We studied efficient replication of tasks in a master-worker distributed computing framework. For a given level of redundancy, we showed that if the service time distribution of a batch is stochastically (decreasing and) convex random variable, a balanced replication of non-overlapping batches of tasks achieves the minimum average job compute time. We then studied the optimum level of redundancy for minimizing average job compute time and maximizing performance predictability. With both exponential-tail and heavy-tail service time distribution of tasks, we showed that the optimum level of redundancy may not agree for the two performance metrics. Thus, we came to the conclusion that if a redundancy level optimizes for predictable performance, then it may increase the average job compute time. Finally, we run experiments on Google cluster trace and showed that a careful planning of redundancy can reduce the average job compute time by an order of magnitude.

ACKNOWLEDGEMENT

This research was supported in part by the NSF awards No. CIF-1717314 and CCF-1559855.

REFERENCES

[1] A. Behrouzi-Far and E. Soljanin, “On the effect of task-to-worker assignment in distributed computing systems with stragglers,” in 2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton). IEEE, 2018, pp. 560–566.

[2] ———, “Data replication for reducing computing time distributed systems with stragglers,” arXiv preprint arXiv:1912.03349, 2019.

[3] A. D. Kshemkalyani and M. Singhal, Distributed computing: principles, algorithms, and systems. Cambridge University Press, 2011.

[4] T. Kraska, A. Talwalkar, J. C. Duchi, R. Griffith, M. J. Franklin, and M. I. Jordan, “Mlibase: A distributed machine-learning system,” in CIDR, vol. 1, 2013, pp. 2–1.

[5] R. Sparks, A. Talwalkar, V. Smith, J. Kottalam, X. Pan, J. Gonzalez, M. J. Franklin, M. I. Jordan, and T. Kraska, “Mli: An api for distributed machine learning,” in 2013 IEEE 13th International Conference on Data Mining. IEEE, 2013, pp. 1187–1192.

[6] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifoski, J. Long, E. J. Shekita, and B.-Y. Su, “Scaling distributed machine learning with the parameter server,” in 11th {USENIX} Symposium on Operating Systems Design and Implementation (OSDI’14), 2014, pp. 583–598.

[7] P. Buchlovsky, D. Budden, D. Grewe, C. Jones, J. Aslanides, F. Besse, A. Brock, A. Clark, S. G. Colmenarejo, A. Pope et al., “Tf-replicator: Distributed machine learning for researchers,” arXiv preprint arXiv:1902.00465, 2019.

[8] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, “Federated optimization: Distributed machine learning for on-device intelligence,” arXiv preprint arXiv:1610.02527, 2016.

[9] J. Choi, “A new parallel matrix multiplication algorithm on distributed-memory concurrent computers,” Concurrency: practice and experience, vol. 10, no. 8, pp. 655–670, 1998.

[10] A. Buluç and J. R. Gilbert, “Parallel sparse matrix-matrix multiplication and indexing: Implementation and experiments,” SIAM Journal on Scientific Computing, vol. 34, no. 4, pp. C170–C191, 2012.

[11] J. Demmel, D. Eliahu, A. Fox, S. Kamil, B. Lipshitz, O. Schwartz, and O. Spillinger, “Communication-optimal parallel recursive rectangular matrix multiplication,” in 2013 IEEE 27th International Symposium on Parallel and Distributed Processing. IEEE, 2013, pp. 261–272.

[12] A. R. Benson and G. Ballard, “A framework for practical parallel fast matrix multiplication,” ACM SIGPLAN Notices, vol. 50, no. 8, pp. 42–53, 2015.

[13] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le et al., “Large scale distributed deep networks,” in Advances in neural information processing systems, 2012, pp. 1222–1231.

[14] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica, “Tune: A research platform for distributed model selection and training,” arXiv preprint arXiv:1807.05118, 2018.

[15] J. Chen, X. Pan, R. Monga, S. Bengio, and R. Jozefowicz, “Revisiting distributed synchronous sgd,” arXiv preprint arXiv:1604.00981, 2016.

[16] R. Anil, G. Pereyra, A. Passos, R. Ormandi, G. E. Dahl, and G. E. Hinton, “Large scale distributed neural network training through online distillation,” arXiv preprint arXiv:1804.03235, 2018.

[17] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein et al., “Distributed optimization and statistical learning via the alternating direction method of multipliers,” Foundations and Trends® in Machine learning, vol. 3, no. 1, pp. 1–122, 2011.

[18] M. Zhu and S. Martínez, “On distributed convex optimization under inequality and equality constraints,” IEEE Transactions on Automatic Control, vol. 57, no. 1, pp. 151–164, 2011.

[19] S. S. Kia, J. Cortés, and S. Martínez, “Distributed convex optimization via continuous-time coordination algorithms with discrete-time communication,” Automatica, vol. 55, pp. 254–264, 2015.

[20] B. Gharesifard and J. Cortés, “Distributed continuous-time convex optimization on weight-balanced digraphs,” IEEE Transactions on Automatic Control, vol. 59, no. 3, pp. 781–786, 2013.

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

[22] G. Ananthanarayanan, A. Ghodsi, S. Shenker, and I. Stoica, “Effective straggler mitigation: Attack of the clones,” in Presented as part of the 10th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI}’13), 2013, pp. 185–198.

[23] M. F. Aktas and E. Soljanin, “Straggler mitigation at scale,” IEEE/ACM Transactions on Networking, vol. 27, no. 6, pp. 2266–2279, 2019.

[24] A. Elser, Reliable distributed systems: technologies, web services, and applications. Springer Science & Business Media, 2005.

[25] Y. Brun, G. Edwards, J. Y. Bang, and N. Medvidovic, “Smart redundancy for distributed computation,” in Distributed Computing Systems (ICDCS), 2011 31st International Conference on. IEEE, 2011, pp. 665–676.

[26] G. Joshi, E. Soljanin, and G. Wornell, “Efficient redundancy techniques for latency reduction in cloud systems,” ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS), vol. 2, no. 2, p. 12, 2017.
The minimum of $N_t$ i.i.d shifted-exponential random variables with rate $\mu$ and shift parameter $\Delta$ has also shifted-exponential distribution, with the following CCDF,

$$F_{T_i}(t) = 1 - \mathbb{I}(t \geq \Delta) \left[ 1 - e^{-N_t \mu(t - \Delta)} \right] \tag{25}$$

Accordingly, $T$ could be written as,

$$T = \max \{ T_1, T_2, \ldots, T_B \} = \Delta + \max \{ T'_1, T'_2, \ldots, T'_B \},$$

where $T'_i \sim \text{Exp}(N_i \mu)$. Therefore, the expected job compute time could be written as,

$$E[T] = \Delta + E[\max \{ T'_1, T'_2, \ldots, T'_B \}].$$

Since $T'_i$s are independent exponential random variables, from Theorem 1 the minimum expected job compute time is achieved by balanced assignment of non-overlapping batches.

C. Proof of Theorem 2

Suppose $T_{ij} \sim \text{pareto}(\sigma, \alpha)$,

$$F_{T_{ij}}(t) = 1 - \mathbb{I}(t \geq \sigma) \left[ 1 - \left( \frac{t}{\sigma} \right)^{-\alpha} \right].$$

It is easy to verify that $T_{ij}$s follow a pareto distribution with,

$$F_{T_{ij}}(t) = 1 - \mathbb{I}(t \geq \sigma) \left[ 1 - \left( \frac{t}{\sigma} \right)^{-N_t \alpha} \right].$$

Therefore, the expected completion time could be written as,

$$E[T] = \max \{ T_1, T_2, \ldots, T_B \} = \sigma + \max \{ T''_1, T''_2, \ldots, T''_B \},$$

where $T''_i \sim \text{Lomax}(N_i \alpha, \sigma)$. Since Lomax random variable is stochastically decreasing and convex, using the same arguments as in Theorem 1, it can be concluded that the minimum expected job compute time, with pareto service time distribution of batches, is achieved by the balanced assignment of non-overlapping batches.

D. Proof of Theorem 3

Suppose $\tau \sim \text{Exp}(\mu)$. With balanced assignment the size of a batch is $N/B$ and the batch service time is exponentially distributed with rate $B\mu/N$,

$$T_{ij} \sim \text{Exp}\left(\frac{B\mu}{N}\right).$$

Note that, in balanced assignment, every group of $N/B$ workers host the same batch. Therefore, the computing time of batch $i$ is,

$$T_i \sim \text{Exp}\left(\frac{B\mu N}{NB} \right) = \text{Exp}(\mu).$$

Thus, the average job compute time in system 1 which is the maximum order statistics of $B$ i.i.d exponential random variables with rate $\mu$, is

$$E[T] = \frac{1}{\mu} H_B. \tag{26}$$
in which $H_B$ is the $B$'th Harmonic number. It can be seen from (26) that smaller $B$ (or higher level of redundancy) achieves lower average compute time, with the minimum being achieved at $B = 1$. Note that $B = 1$ means that every worker host the compute job, i.e. the maximum diversity and the minimum parallelism.

E. Proof of Theorem 2

To prove this theorem we show that the function in (18) is monotonically decreasing for all $B$. To show a contradiction, suppose this function is increasing, for some values of $B$, 

$$\frac{H(B,2)}{H^2(B,1)} \leq \frac{H(B+1,2)}{H^2(B+1,1)}, \quad \exists B \in \mathbb{N},$$

or equivalently,

$$\frac{H(B,2)}{H^2(B,1)} \leq \frac{H(B,2) + \frac{1}{(B+1)^2} + \frac{2}{B+1}H(B,1)}{H^2(B,1)}, \quad \exists B \in \mathbb{N}.$$ 

By further simplifications,

$$2H(B,1)H(B,2) \leq \frac{1}{B+1} \left( H^2(B,1) - H(B,2) \right),$$

and,

$$2(B+1) \leq \frac{H(B,1)}{H(B,2)} - \frac{1}{H(B,1)},$$

Knowing that,

$$\frac{H(B,1)}{H(B,2)} - \frac{1}{H(B,1)} < 1,$$

we have,

$$2(B+1) \leq 1,$$

and $B \leq -1/2$, which is a contradiction with $\exists B \in \mathbb{N}$. Therefore, the function in (18) is monotonically decreasing. Consequently, the minimum coefficient of variations is achieved at full parallelism.

F. Proof of Theorem 5

Suppose $\tau \sim SExp(\Delta, \mu)$. According to the size-dependent service time model, the service time of a worker with a batch of size $N/B$ will be the random variable $T_{ij} = \frac{N}{B}\tau$. Thus, the CDF of the batch compute time, that is available at $N/B$ workers, could be written as,

$$F_{\tau}(t) = \Pr(T_{ij} > t)$$

$$= \Pr\{\min\{T_{i1}, T_{i2}, \ldots , T_{iN/B}\} > t\},$$

$$= \prod_{j=1}^{N/B} \Pr\{T_{ij} > t\}$$

$$= \prod_{j=1}^{N/B} \Pr\{\tau > Bt/N\}$$

$$= \left[ \Pr\{\tau > Bt/N\} \right]^{N/B}$$

$$= \left[ F_{\tau}(Bt/N) \right]^{N/B}.$$ 

where,

$$F_{\tau}(Bt/N) = 1 - \mathbb{1}(Bt/N \geq \Delta) \left[ 1 - e^{-\mu(Bt/N - \Delta)} \right],$$

and with further simplifications,

$$\bar{F}_{\tau}(Bt/N) = 1 - \mathbb{1}(t \geq N\Delta/B) \left[ 1 - e^{-B\mu/N(t-N\Delta/B)} \right].$$

Therefore,

$$\bar{F}_{\tau}(t) = 1 - \mathbb{1}(t \geq N\Delta/B) \left[ 1 - e^{-\mu(t-N\Delta/B)} \right].$$

or equivalently,

$$\bar{F}_{\tau}(t) = 1 - \mathbb{1}(t \geq N\Delta/B) \left[ 1 - e^{-\mu(t-N\Delta/B)} \right].$$

In other words,

$$T_i \sim SExp(N\Delta/B, \mu).$$

Note that, the CDF of $T_i$ is

$$F_{\tau}(t) = \mathbb{1}(t \geq N\Delta/B) \left[ 1 - e^{-\mu(t-N\Delta/B)} \right].$$

Accordingly, the CDF of the job compute time in system [1], which is the maximum of $B$ i.i.d shifted exponential random variables, could be written as,

$$F_T(t) = \Pr\{T < t\} = \mathbb{1}(\max\{T_1, T_2, \ldots , T_B\} < t\),$$

$$= \prod_{i=1}^{B} \Pr\{T_i < t\},$$

$$= \left[ F_{\tau}(t) \right]^B.$$ 

Therefore,

$$F_T(t) = \mathbb{1}(t \geq N\Delta/B) \left[ 1 - e^{-\mu(t-N\Delta/B)} \right]^B.$$ 

From that, the average job compute time is,

$$E[T] = \int_{0}^{+\infty} F_T(t)dt,$$

$$= \int_{0}^{N\Delta/B} dt + \int_{N\Delta/B}^{+\infty} \left[ 1 - \left( 1 - e^{-\mu(t-N\Delta/B)} \right)^B \right] dt,$$

$$= \frac{N\Delta}{B} + \frac{1}{\mu}H(B,1).$$

The function (33) is not monotonic in $B$. Therefore, to find the optimum $B$, one may solve the following discrete unconstrained optimization problem,

$$\min_{B \in F_B} \frac{N\Delta}{B} + \frac{1}{\mu}H(B,1),$$

where $F_B$ is the set of all feasible values for $B$.

G. Proof of Theorem 6

Let’s evaluate the average completion time (19) at the two smallest operating points.

$$B = 1: \quad E[T] = N\Delta + 1/\mu$$

$$B = 2: \quad E[T] = N\Delta/2 + 3/2\mu.$$ 

For the average job compute time to be initially increasing in $B$ the function (19) should have smaller value at $B = 1$ than
\( B = 2 \), i.e. \( \Delta \mu < 1/N \). Evaluating this function at the two largest operating points,

\[
B = N/2 : \quad \mathbb{E}[T] = 2\Delta + H(N/2,1)/\mu \\
B = N : \quad \mathbb{E}[T] = \Delta + H(N,1)/\mu ,
\]

reveals that for it to ends increasing the product \( \Delta \mu \) should satisfy,

\[
\Delta \mu < H(N,1) - H(N/2,1), \quad \text{(34)}
\]
or equivalently,

\[
\Delta \mu < \sum_{k=N/2+1}^{N} 1/k. \quad \text{(35)}
\]

Now given that \( 1/N < \sum_{k=N/2+1}^{N} 1/k, \) if \( \Delta \mu < 1/N \) the function (19) is monotonically increasing, which makes full diversity the optimum operating point. If \( \Delta \mu > \sum_{k=N/2+1}^{N} 1/k \) then the function (19) is monotonically decreasing and full parallelism is the optimum operating point. Finally, if \( 1/N \leq \Delta \mu \leq \sum_{k=N/2+1}^{N} 1/k \) then the function (19) reaches a minimum point in neither of the diversity-parallelism spectrum.

**H. Proof of Corollary 2**

Using the approximation of the harmonic number \( H_B = \log B + \gamma \), the average completion time (19) can be approximated by,

\[
f(B) := \frac{N\Delta}{B} + \frac{1}{\mu} \log B + \gamma/\mu, \quad \text{(36)}
\]

where \( \gamma \) is the Euler-Mascheroni constant. The minimum of (36) occurs at \( B_{\text{min}} = N\Delta \mu \). In order to find the minimum of the function (19), we need to find the value of \( B \) which results in the minimum deviation from the value of function (36) at \( B_{\text{min}} \), which is zero. We call this deviation \( \delta \mathbb{E}[T] \). The derivative of \( f(B) \) is,

\[
f'(B) = \frac{1}{\mu B} - \frac{N\Delta}{B^2}. \quad \text{(37)}
\]

Therefore,

\[
\delta \mathbb{E}[T] = (B - B_{\text{min}})[f'(B) - f'(B_{\text{min}})] ,
\]

\[
= (B - N\Delta \mu) \left[ \frac{1}{\mu B} - \frac{N\Delta}{B^2} \right] ,
\]

\[
= \frac{1}{\mu} \left( 1 - \frac{N\Delta \mu}{B} \right)^2. \quad \text{(38)}
\]

Accordingly a value of \( B \) which minimizes \( (1 - N\Delta \mu/B)^2 \), or equivalently \( |B - N\Delta \mu| \), is the optimum operating point.

**I. Proof of Theorem 7**

Let’s evaluate the coefficient of variations (21) at the two smallest operating points,

\[
B = 1 : \quad \text{CoV}[T] = 1/\Delta \mu N \\
B = 2 : \quad \text{CoV}[T] = \sqrt{5}/(\Delta \mu N + 3).
\]

For the coefficients of variations to be initially increasing in \( B \) the function (21) should have smaller value at \( B = 1 \) than at \( B = 2 \), i.e. \( \Delta \mu < 3/(\sqrt{5} - 1)N \). Evaluating the same function at the two largest operating points,

\[
B = N/2 : \quad \text{CoV}[T] = \sqrt{H(N/2,2)/(2\Delta \mu + H(N/2,1))} \\
B = N : \quad \text{CoV}[T] = \sqrt{H(N,2)/(\Delta \mu + H(N,1))},
\]

reveals that for it to ends increasing the product \( \Delta \mu \) should satisfy,

\[
\Delta \mu > \frac{H(N,1)\sqrt{1/H(N,2) - H(N,2)/1/(H(N,2))}}{2\sqrt{H(N,2) - H(N,2)/1/(H(N,2))}}. \quad \text{(39)}
\]

According to the limiting behaviour

\[
\lim_{N \to \infty} \frac{H(N,2)}{H(N,2)} = 1, \quad \text{(40)}
\]

we can approximate the RHS of (39) as,

\[
\lim_{N \to \infty} \frac{H(N,1)\sqrt{1/H(N,2) - H(N,2)/1/(H(N,2))}}{2\sqrt{H(N,2) - H(N,2)/1/(H(N,2))}} = H(N,1) - H(N,2). \quad \text{(41)}
\]

Further, using the following series expansion around infinity

\[
H(N,1) - H(N,2) = \log 2 - \frac{1}{2N} + O(N^{-2}) \quad \text{(42)}
\]

it is easy to verify that \( \forall N \in \{5, 6, 7, \ldots \} \)

\[
\frac{3}{(\sqrt{5} - 1)N} < H(N,1) - H(N,2). \quad \text{(43)}
\]

Therefore, for any integer \( N > 4 \), the function (21) is monotonically decreasing when \( \Delta \mu < 3/(\sqrt{5} - 1)N \) and thus full parallelism minimizes the coefficient of variations. It is monotonically increasing when \( \Delta \mu > \frac{H(N,1)\sqrt{1/H(N,2) - H(N,2)/1/(H(N,2))}}{2\sqrt{H(N,2) - H(N,2)/1/(H(N,2))}} \) and therefore full diversity minimized the coefficient of variations. Finally, when the product \( \Delta \mu \) is between the two bounds, the function starts increasing and ends decreasing. In this case, the minimum coefficient of variations occur either at full diversity of at full parallelism.

**J. Proof of Corollary 3**

From theorem 7 with \( 3/(\sqrt{5} - 1)N \leq \Delta \mu \leq \frac{H(N,1)\sqrt{1/H(N,2) - H(N,2)/1/(H(N,2))}}{2\sqrt{H(N,2) - H(N,2)/1/(H(N,2))}} \), the coefficient of variations is minimum at one of the two ends of the diversity-parallelism spectrum. If the value of the function is smaller at \( B = 1 \) than \( B = N \) then full diversity is optimal. For that, the product \( \Delta \mu \) should satisfy,

\[
\Delta \mu < \frac{H(N,1)}{N\sqrt{H(N,2)}}. \quad \text{(44)}
\]

Otherwise full parallelism is optimum. The RHS of (44) can be approximated by \( H(N,1)/N\sqrt{H(N,2)} \). This approximation can be interpreted in terms of the derivatives of digamma function,

\[
\frac{H(N,1)}{\sqrt{H(N,2)}} = \frac{\gamma(0)(N + 1)}{\sqrt{\pi^2/6 - \psi(0)(N + 1)}}. \quad \text{(45)}
\]
where $\psi^{(i)}(.)$ is the $i$th derivative of the digamma function. The RHS of (45) can be expanded around $\infty$ as,
\[
\frac{\gamma \psi^{(0)}(N + 1)}{\sqrt{\pi^2/6 - \psi^{(1)}(N + 1)}} = \frac{\sqrt{6 (\gamma + \log N)}}{\pi} + \sqrt{\frac{3}{2} 6 \gamma + 6 \log N + \pi^2}{\pi^3 N} + O \left( \frac{\log N}{N^2} \right).
\]
Therefore, for large values of $N$, the LHS of (45) can be well approximated by,
\[
\frac{H_{(N,1)}}{\sqrt{H_{(N,2)}}} \approx \frac{\sqrt{6 (\gamma + \log N)}}{\pi} + \sqrt{\frac{3}{2} 6 \gamma + 6 \log N + \pi^2}{\pi^3 N}
\]
It is easy to verify that (47) is increasing $\forall N \in \mathbb{N}$. Moreover, $\forall N \in \{12, 13, \ldots \}$
\[
\frac{3}{\sqrt{5} - 1} < \frac{\sqrt{6 (\gamma + \log N)}}{\pi} + \sqrt{\frac{3}{2} 6 \gamma + 6 \log N + \pi^2}{\pi^3 N}.
\]
Thus,
\[
\frac{3}{(\sqrt{5} - 1) N} < \frac{H_{(N,1)}}{N(\sqrt{H_{(N,2)}} - 1)}, \quad \forall N > 11,
\]
and for any value of $\Delta \mu$ that satisfies
\[
\frac{3}{(\sqrt{5} - 1) N} < \Delta \mu < \frac{H_{(N,1)}}{N(\sqrt{H_{(N,2)}} - 1)}
\]
CoV[$T$] is minimized at full parallelism. Next we prove that $H_{(N,1)}/\sqrt{H_{(N,2)}}$ is smaller than $H_{(N,1)} - H_{(N,2,1)}$ for $N > 4$. It is easy to verify that
\[
\frac{N - 1}{N/2 + j} - \frac{1}{j} > 0, \quad \forall N > 4, j \in \mathbb{N}.
\]
Therefore,
\[
(N - 1) \sum_{k=N/2+1}^{N/2} \frac{1}{k} - \sum_{k=1}^{N/2} \frac{1}{k} > 0, \quad \forall N > 4, j \in \mathbb{N}.
\]
Thus we can write,
\[
1 + \frac{\sum_{k=N/2+1}^{N} 1/k}{\sum_{k=1}^{N/2} 1/k} > 1 + \frac{1}{N - 1}, \quad \forall N > 4.
\]
Accordingly,
\[
H_{(N,1)} < N \left[ H_{(N,1)} - H_{(N,2,1)} \right], \quad \forall N > 4,
\]
Since
\[
\sqrt{H_{(N,2)}} \geq 1, \quad \forall N \in \mathbb{N},
\]
we can update the bound in (54) to,
\[
H_{(N,1)} < N \left[ H_{(N,1)} - H_{(N,2,1)} \right] \sqrt{H_{(N,2)}}, \quad \forall N > 4.
\]
Consequently,
\[
\frac{H_{(N,1)}}{N \sqrt{H_{(N,2)}}} < H_{(N,1)} - H_{(N,2,1)},
\]
and thus for any value of $\Delta \mu$ that satisfies,
\[
\frac{H_{(N,1)}}{N \sqrt{H_{(N,2)}}} < \Delta \mu < H_{(N,1)} - H_{(N,2,1)}, \quad \forall N > 4.
\]
CoV[$T$] is minimised at full diversity.

**K. Proof of Corollary**

Let’s look at the asymptotic behaviour of the distance of $H_{(N,1)}/\sqrt{H_{(N,2),1}}$ to the two bounds in corollary
\[
\lim_{N \to \infty} \frac{H_{(N,1)} - H_{(N,2,1)} - H_{(N,1)}/\sqrt{H_{(N,2),1}}}{\sqrt{H_{(N,2),1}}} = \lim_{N \to \infty} \frac{\log 2 - H_{(N,1)}/\sqrt{H_{(N,2),1}}}{\sqrt{H_{(N,2),1}} - 3/(\sqrt{5} - 1) N} = \infty.
\]
Thus we have,
\[
\Delta \mu \in \left( \frac{H_{(N,1)}}{N \sqrt{H_{(N,2)}}}, H_{(N,1)} - H_{(N,2,1)} \right), \quad \text{as } N \to \infty.
\]
Therefore, from corollary the minimum CoV[$T$] occurs at full diversity.

**L. Proof of Theorem**

Suppose $\tau \sim$ Pareto($\sigma, \alpha$). The CCDF of the service time of a batch with size $N/B$ is,
\[
\Pr(T_{ij} > t) = \Pr(\tau > Bt/N),
\]
\[
= 1 - 1 \left( t \geq N \sigma / B \right) \left[ 1 - \left( \frac{Bt}{N \sigma} \right)^{-\alpha} \right].
\]
And the CCDF of $T_i$ is,
\[
\Pr(T_i > t) = \left[ \Pr(T_{ij} > t) \right]^{N/B},
\]
\[
= 1 - 1 \left( t \geq N \sigma / B \right) \left[ 1 - \left( \frac{Bt}{N \sigma} \right)^{-N \alpha / B} \right].
\]
Hence, $T_i \sim$ Pareto($N \sigma / B, N \alpha / B$). From (92), $E[T]$, which is the maximum order statistics of $B$ random variables following Pareto($N \sigma / B, N \alpha / B$), can be written as,
\[
E[T] = \frac{N \sigma \Gamma(B + 1) \Gamma(1 - B / N \alpha)}{B \Gamma(B + 1 - B / N \alpha)},
\]
which completes the proof.

**M. Proof of Theorem**

For the objective function (22) to be initially decreasing in $B$, it should be smaller at $B = 2$ than at $B = 1$.
\[
B = 1: \quad \mathbb{E}[T] = N \sigma \frac{\Gamma(2) \Gamma(1 - 1 / N \alpha)}{\Gamma(2 - 1 / N \alpha)},
\]
\[
B = 2: \quad \mathbb{E}[T] = \frac{N \sigma \Gamma(3) \Gamma(1 - 2 / N \alpha)}{\Gamma(3 - 2 / N \alpha)}.
\]
In other words,
\[
\frac{\Gamma(1 - 2 / N \alpha) \Gamma(2 - 1 / N \alpha)}{\Gamma(1 / N \alpha) \Gamma(3 - 2 / N \alpha)} > 1.
\]
From the properties of Gamma function, it can be verified that,
\[
\frac{\Gamma(1 - 2 / N \alpha) \Gamma(2 - 1 / N \alpha)}{\Gamma(1 / N \alpha) \Gamma(3 - 2 / N \alpha)} = \frac{1}{2(1 - 2 / N \alpha)}.
\]
By further simplification, it yields to $\alpha > 4/N$. Remember, we assumed that $N > 4$. Thus $4/N < 1$ and with $\alpha > 1$ the objective function in (22) is initially decreasing in $B$. Likewise, for the function (22) to ends increasing it should be smaller at $B = N/2$ than at $B = N$.

$$B = N/2: \quad \mathbb{E}[T] = 2\sigma \frac{\Gamma(N/2 + 1)\Gamma(1 - 1/2\alpha)}{\Gamma(N/2 + 1 - 1/2\alpha)},$$

$$B = N: \quad \mathbb{E}[T] = \sigma \frac{\Gamma(N + 1)\Gamma(1 - 1/\alpha)}{\Gamma(N + 1 - 1/2\alpha)}.$$ 

In other words,

$$\frac{\Gamma(N + 1)\Gamma(1 - 1/\alpha)}{\Gamma(N + 1 - 1/2\alpha)} > \frac{2\Gamma(N/2 + 1)\Gamma(1 - 1/2\alpha)}{\Gamma(N/2 + 1 - 1/2\alpha)}. \quad (64)$$

Substituting $\Gamma(N + 1) = N\Gamma(N)$, the inequality (64) can be rewritten as

$$\frac{\Gamma(N)\Gamma(1 - 1/\alpha)}{\Gamma(N + 1 - 1/2\alpha)} > \frac{\Gamma(N/2)\Gamma(1 - 1/2\alpha)}{\Gamma(N/2 + 1 - 1/2\alpha)}. \quad (65)$$

According to the asymptotic behaviour of Gamma function

$$\lim_{N \to \infty} \frac{\Gamma(N + x)}{\Gamma(N)N^x} = 1, \quad (66)$$

for large $N$ the inequality (65) can be written as

$$\frac{\Gamma(1 - 1/\alpha)}{N^{1-1/\alpha}} > \frac{\Gamma(1 - 1/2\alpha)}{(N/2)^{1-1/2\alpha}}. \quad (67)$$

Furthermore, it can be written that

$$\frac{\Gamma(1 - 1/2\alpha)}{\Gamma(1 - 1/\alpha)} = \frac{2^{1/\alpha}\sqrt{\pi}}{\Gamma(1/2)^{1-1/2\alpha}}. \quad (68)$$

Accordingly, for large $N$ the inequality (65) is written as

$$\Gamma(1/2 - 1/2\alpha) > \sqrt{\pi}N^{-1/2\alpha}2^{1+1/2\alpha}. \quad (69)$$

Since $1/2 - 1/2\alpha < 1/2$, we can further simplify (69) by using the Laurent expansion of Gamma function around 0,

$$\Gamma(x) = \frac{1}{x} - \gamma + \frac{\pi^2 + 6\gamma^2}{12}x + O(x^2) \quad (70)$$

Therefore, (64) can be rewritten as,

$$\frac{4\alpha^2 + (\alpha - 1)^2}{2\alpha(\alpha - 1)} - \sqrt{\pi}N^{-1/2\alpha}2^{1+1/2\alpha} - 0.58 > 0. \quad (71)$$

It is easy to verify that in (69) the RHS is a decreasing function of $\alpha$ and the LHS is an increasing function of $\alpha$, for $\alpha > 1$. That means,

$$\Gamma(1/2 - 1/2\alpha) > \sqrt{\pi}N^{-1/2\alpha}2^{1+1/2\alpha}, \quad \forall 1 < \alpha < \alpha^*, \quad (72)$$

and,

$$\Gamma(1/2 - 1/2\alpha) < \sqrt{\pi}N^{-1/2\alpha}2^{1+1/2\alpha}, \quad \forall \alpha \geq \alpha^*, \quad (73)$$

where $\alpha^*$ is the solution of

$$\frac{4\alpha^2 + (\alpha - 1)^2}{2\alpha(\alpha - 1)} - \sqrt{\pi}N^{-1/2\alpha}2^{1+1/2\alpha} - 0.58 = 0. \quad (74)$$

### N. Proof of Lemma 6

From Theorem 8 the distribution of the computing time of batch $i$ follows Pareto ($N\sigma/B, N\alpha/B$). The covariance of order statistics of $n$ pareto($\sigma', \alpha'$) random variable is given in (92) as,

$$\text{Cov}[X_{k_1:n}, X_{k_2:n}] = \sigma'^2 \frac{n!}{(n-k_2)!} \frac{\Gamma(n-k_1 + 1 - 2/\alpha')}{\Gamma(n - 2/\alpha')} \times \frac{\Gamma(n-k_2 + 1 - 1/\alpha')}{\Gamma(n-k_1 + 1 - 1/\alpha')} - E[X_{k_1:n}]E[X_{k_2:n}].$$

By setting $k_1 = k_2 = n$, the variance of the maximum order statistics is,

$$\text{Var}[X_{n:n}] = \sigma'^2 \frac{n!}{(n-1)!} \frac{\Gamma(n - 2/\alpha')}{\Gamma(n + 1 - 2/\alpha')} - E[X_{n:n}]^2. \quad (75)$$

Substituting (61), $n = B$, $\sigma' = N\sigma/B$ and $\alpha' = N\alpha/B$,

$$\text{Var}(T) = \left(\frac{N\sigma}{B}\right)^2 \frac{\Gamma(B + 1)\Gamma(1 - 2B/N\alpha)}{\Gamma(B + 1 - 2B/N\alpha)} \times \left(\frac{N\sigma}{B}\frac{\Gamma(B + 1)\Gamma(1 - B/N\alpha)}{\Gamma(B + 1 - B/N\alpha)}\right)^2. \quad (76)$$

Consequently,

$$\text{CoV}[T] = \frac{\sqrt{\text{Var}[T]}}{\mathbb{E}[T]} = \frac{\sqrt{\frac{\Gamma(B + 1 - B/N\alpha)\Gamma(1 - 2B/N\alpha)}{\Gamma(B + 1 - 2B/N\alpha)\Gamma(1 - B/N\alpha) - 1}}}{\frac{\Gamma(B + 1)\Gamma(1 - 2B/N\alpha)}{\Gamma(B + 1 - B/N\alpha)} - 1. \quad (77)$$

### O. Proof of Theorem 10

We define the two ratio

$$Q_1(B) = \frac{\Gamma(B + 1 - B/N\alpha)}{\Gamma(B + 1 - 2B/N\alpha)}, \quad (78)$$

$$Q_2(B) = \frac{\Gamma(1 - 2B/N\alpha)}{\Gamma(1 - B/N\alpha)}. \quad (79)$$

Let’s define

$$Q'_1(B) = \frac{\Gamma(1 + B(1 - 1/N\alpha))}{\Gamma(1 + B(1 - 2/N\alpha))}. \quad (80)$$

as the continuous version of $Q_1(B)$. The derivative of $Q'_1(B)$ is,

$$\frac{dQ'_1(B)}{dB} = \frac{Q'_1(B)}{N\alpha} \left[ (N\alpha - 1)\psi(1 + B(1 - 1/N\alpha)) - (N\alpha - 2)\psi(1 + B(1 - 2/N\alpha)) \right] \quad (81)$$

which can be rewritten in the form of,

$$\frac{dQ'_1(B)}{dB} = \frac{Q'_1(B)}{N\alpha} \left[ (N\alpha - 1)\left(\psi(1 + B(1 - 1/N\alpha)) - \psi(1 + B(1 - 2/N\alpha)) \right) \right] + \psi(1 + B(1 - 2/N\alpha)). \quad (82)$$

Here $\psi(.)$ is the digamma function, which is increasing in the positive real domain. Therefore,

$$(N\alpha - 1)\left(\psi(1 + B(1 - 1/N\alpha)) - \psi(1 + B(1 - 2/N\alpha)) \right) > 0. \quad (83)$$
For the set of parameters in our system \((B \geq 1, \alpha > 2, N > 4)\),
\[
\psi(1 + B(1 - 2/N\alpha)) \geq \psi(1.75) > 0. \tag{84}
\]
Furthermore,
\[
\psi(1/2 - B/N\alpha) + \log 4 < \psi(1/2) + \log 4
\approx -0.57. \tag{87}
\]
For the set of parameters in our system \((B \geq 1, \alpha > 2, N > 4)\),
\[
\psi(1 + B(1 - 2/N\alpha)) \geq \psi(1.75) > 0. \tag{84}
\]
Further, it is easy to verify that \(Q'_1(B) > 0\). Consequently, \(Q'_1(B)\) is an increasing function, for our set of parameters. Given that \(Q'_1(B)\) is smooth, we can argue that \(Q_1(B)\) is also increasing. Likewise, we define the continuous counterpart of \(Q_2(B)\) as
\[
Q'_2(B) = \frac{\Gamma(1 - 2B/N\alpha)}{\Gamma(1 - B/N\alpha)} = \frac{2^{-2B/N\alpha}}{\sqrt{\pi}} \Gamma(1/2 - B/N\alpha). \tag{85}
\]
We can write,
\[
\frac{dQ'_2(B)}{dB} = -\left[\psi(1/2 - B/N\alpha) + \log 4\right] \times \frac{2^{-2B/N\alpha} \Gamma(1/2 - B/N\alpha)}{N\alpha \sqrt{\pi}}. \tag{86}
\]
Accordingly, it is true that \(dQ'_2(B)/dB > 0\) and thus \(Q'_2(B)\) is increasing. Due to the smoothness of \(Q'_2(B)\), we can argue that \(Q_2(B)\) is also increasing. Finally, with both \(Q_1(B)\) and \(Q_2(B)\) being increasing, we can say that the function in \([24]\) is increasing as well. Hence, the minimum coefficient of variations of completion time is achieved at minimum \(B\), i.e. full diversity.