Gradient Coding

February 3, 2021

1 Preliminaries

Given data **D** = {(x$_1$,y$_1$), (x$_2$,y$_2$),....,(x$_d$,y$_d$)}, where each $(x, y) \in \mathcal{R}^p \times R$, several machine learning tasks aim to solve the following problem:

$$\beta^* = argmin \sum_{i=1}^{d} L(\beta; x_i, y_i) + \lambda R(\beta) \tag{1}$$

where *L(·)* is a task-specific loss function, and *R(·)* is a regularization function. Typically, this optimization problem can be solved using gradient-based approaches. Let

$$g = \sum_{i=1}^{d} \nabla L(\beta^{(t)}; x_i, y_i) \tag{2}$$

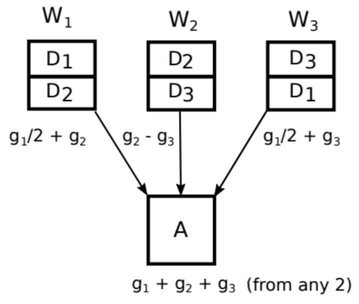
be the gradient of the loss at the current model β. Then the updates to the model are of the form:

$$\beta^{(t+1)} = h_R(\beta^{(t)}, g) \tag{3}$$

where h_{R} is a gradient-based optimizer, which also depends on $R(\,\cdot\,)$. Several methods such as gradient descent, accelerated gradient, conditional gradient (Frank-Wolfe), proximal methods, LBFGS, and bundle methods fit in this framework. However, if the number of samples, d, is large, a computational bottleneck in the above update step is the computation of the gradient, g, whose computation can be distributed.

2 Problem Setup and Notations (changed)

In this work, we try to find a novel coding theoretic framework for mitigating stragglers in distributed learning, particularly synchronous gradient descent. Consider a setup with n workers out of which atmost s workers can straggle. The baseline vanilla problem for recovering the entire gradient at each step of the gradient descent algorithm is considered in [1]. It is important to note the goal is not only to minimize the computation load at each worker but also to minimize the communication costs between the master and workers i.e. for any scheme to be efficient, every worker should transmit some linear combination of the gradients of datasets assigned to it at each step. Here is an example to illustrate the same: Consider 3 workers W$_1$, W$_2$, W$_3$ out of which any 1 worker may straggle. One of the possible schemes for this problem is shown in figure 1. Each worker is assigned to compute two gradients on the two examples they have for this round. The main coding theoretic question we investigate is how to design these linear combinations so that any two (or any fixed number generally) contain the $g_1 + g_2 + g_3$ vector in their span. In our example, in Figure 1, W_1 sends $\frac{1}{2}g_1 + g_2$, W_2 sends g_2 - g_3 and W_3 sends $\frac{1}{2}g_1 + g_3$. The reader can verify that A can obtain the vector $g_1 + g_2 + g_3$ from any two out of these three vectors. For instance, $g_1 + g_2 + g_3 = 2(\frac{1}{2}g_1 + g_2)$ - $(g_2$ - $g_3)$. This idea is called gradient coding.



(b) Gradient coding: The vector $g_1 + g_2 + g_3$ is in the span of *any two* out of the vectors $g_1/2 + g2$, $g_2 - g_3$ and $g_1/2 + g_3$.

Figure 1: Example

Some Notation:
$n \rightarrow$ Total number of workers
$s \rightarrow$ Number of stragglers
$k \rightarrow$ Number of parts data is divided
$g_{\mathrm{i}} \rightarrow$ Gradient of dataset $\mathrm{d_i}$
$B \rightarrow n \times k$ incidence matrix(b_{ij} is the (i,j)$^{\mathrm{th}}$ entry)
$b_{\mathrm{i}} \rightarrow$ i$^{\mathrm{th}}$ row of matrix B
Each worker transmits $b_i g$ to the master.

Condition 1: Consider any scheme robust to any s stragglers, given n workers (with $s<n$). Then we require that for every subset $I \subsetneq [n]$, $\|I\| = n$ - s:

$$1_{1\times k} \in span\{b_i | i \in I\} \tag{4}$$

Theorem 1: Consider any scheme robust to any s stragglers, given n workers (with $s<n$) and k partitions. Then, if all rows of B have the same number of non-zeros, we must have [1]:

$$||b_i|| \geqslant \frac{k}{n}(s+1) \tag{5}$$

Proof:
Consider the incidence matrix(B) of any scheme satisfying *Condition 1.* Observe that each column

of this matrix should have atleast $s+1$ non-zero elements. Therefore, there will be atleast $k(s+1)$ non-zero elements in B. So, atleast one row(meaning one worker) of this matrix should have,

$$\frac{\text{Load}}{\text{Worker}} \geqslant \frac{k}{n}(s+1) \tag{6}$$

A couple of schemes that can achieve this bound have been discussed in [1].
In our project we looked at a modified problem statement. In the baseline problem we were required to compute the gradient of the entire dataset at each step/iteration. Instead, now we look at how the load per worker varies if we compute the gradient of only a fraction of the dataset. The fraction (α) will be another input to the coding scheme along with n and s.

We divide the dataset is divided into k parts and l denotes the number of parts whose gradient we compute in every iteration. We denote the load as the number of data-parts whose gradient is computed by each worker.

$$l = \alpha k \tag{7}$$

Condition 2: Consider any scheme robust to any s stragglers, given n workers (with $s<n$). Then we require that for every subset $I \subsetneq [n]$, $\|I\| = n - s$:

$$v \in \text{span}\{b_i \mid i \in I\} \tag{8}$$

for some vector $v \in \{0,1\}^{\text{n}}$, with $||v|| \geqslant l$ We define load per worker as the number of parts of the data whose gradients are computed by each worker.

The following lemma gives some bound on the number of workers who are given one part of the data.

Lemma 1. *If any scheme robust to s stragglers with n workers can recover l gradients with each part of the data being given y distinct workers for gradient computation, then it satisfies the following condition.*

$$\frac{\binom{s}{y}}{\binom{n}{y}} \leq 1 - \alpha \tag{9}$$

Proof. We prove this as follows. We consider all possible subsets of cardinality s of set of n workers given to us, we know that there are $\binom{n}{s}$ such sub-sets. Let us denote these subsets by S_i for $i \in [\binom{n}{s}]$. Now consider any data-part $j \in [k]$ and consider the set of workers it is assigned to by W_j, clearly whose cardinality is y. Since we know that each subset of $(n-s)$ workers should have access to at least α fraction of the dataset, each subset S_i can have at most $k(1-\alpha)$ subsets W_j such that $W_j \subseteq S_i$.

Let us consider the denote $k_i = |\{W_j | j \in [n]; W_j \subseteq S_i\}|$.We attempt to bound $\sum_{i\in[\binom{n}{s}]} k_i$.

As argued above $\sum_{i\in[\binom{n}{s}]} k_i \leq \binom{n}{s}k(1-\alpha)$. However each set W_j is present in exactly $\binom{n-y}{s-y}$ subsets S_i for some $i \in [\binom{n}{s}]$. Thus, $\sum_{i\in[\binom{n}{s}]} k_i = k\binom{n-y}{s-y}$.

Thus, we get $k\binom{n-y}{s-y} \leq \binom{n}{s}k(1-\alpha)$ implying the condition in 22.

□

Lemma 2. *If any scheme robust to s stragglers with n workers can recover l gradients with part i of the data being given y_i distinct workers for gradient computation, then it satisfies the following condition*

$$\sum_{i=1}^{k} \binom{n-y_i}{n-s} \leq \binom{n}{s} k(1-\alpha) \tag{10}$$

Note: A slight change in the the the theorem where we relaxed the assumption on the number of parts being same as the number of workers. Also in the previous part, we assumed each part is assigned to y workers implying the number of parts assigned to every worker is $\frac{ky}{n}$ under a balanced We also propose a slight corollary to this theorem where the same bound also holds if the system is unbalanced i.e. each part is assigned to a different number of workers.

Claim 1. *The following claim holds true for any set of positive integers $\{a_i\}_{1\leq i\leq t}$. Define $a = \lfloor\frac{\sum_{i=1}^{t} a_i}{t}\rfloor$ and $t_1 = (a+1)t - \sum_{i=1}^{t} a_i$.Then we have the following*
$\sum_{i=1}^{t}\binom{a_i}{r} > t_1.\binom{a}{r} + (t-t_1).\binom{a+1}{r}$. Note that $\sum a_i = t_1.a + (t-t_1)(a+1)$

Proof.

$$\binom{x+m_1}{r} - \binom{x}{r} = \sum_{i=1}^{m_1}\binom{x+i-1}{r-1}$$

$$\binom{x+1}{r} - \binom{x-m_2+1}{r} = \sum_{i=1}^{m_2}\binom{x-i+1}{r-1}$$

These follow from $\binom{n}{r} + \binom{n}{r+1} = \binom{n+1}{r+1}$. Since $\binom{x+i-1}{r-1} \geq \binom{x-j+1}{r-1}$ for any $0 \leq i \leq m_1$, $0 \leq j \leq m_2$, we can say $\frac{\binom{x+m_1+1}{r}-\binom{x+1}{r}}{m_1} \geq \frac{\binom{x+m_1}{r}-\binom{x}{r}}{m_1} \geq \frac{\binom{x+1}{r}-\binom{x-m_2+1}{r}}{m_2} \geq \frac{\binom{x}{r}-\binom{x-m_2}{r}}{m_2}$

Choose the list I as follows: $\{i \in [t] : a_i > a+1\}$ and list J as $\{i \in [t] : a_i < a\}$. Now choose a partition of I s.t $I = I_1 \cup I_2$ and $I_1 \cap I_2 = \Phi$ and J s.t $J = J_1 \cup J_2$ and $J_1 \cap J_2 = \Phi$ s.t $|I_1 \cup J_1 \cup \{i \in [t] : a_i = a+1\}| = t - t_1$. This would imply $|I_2 \cup J_2 \cup \{i \in [t] : a_i = a\}| = t_1$

Now denote $k_{min} = \min\limits_{m_1>1} \frac{\binom{x+m_1}{r}-\binom{x}{r}}{m_1}$ and $k_{max} = \max\limits_{m_2>1} \frac{\binom{x+1}{r}-\binom{x-m_2+1}{r}}{m_2}$

Thus,

$$\begin{aligned}
&\sum_{i\in I_1}\Big[\binom{a_i}{r} - \binom{a+1}{r}\Big] + \sum_{i\in I_2}\Big[\binom{a_i}{r} - \binom{a}{r}\Big] \\
\overset{(a)}{\geq}& \sum_{i\in I_1}\Big[\binom{a_i-1}{r} - \binom{a}{r}\Big] + \sum_{i\in I_2}\Big[\binom{a_i}{r} - \binom{a}{r}\Big] \\
\geq& k_{min}\Big(\sum_{i\in I_1}(a_i-a-1) + \sum_{i\in I_2}(a_i-a)\Big)
\end{aligned}$$

$$\begin{aligned}
&\sum_{i\in J_1}[\binom{a+1}{r}-\binom{a_i}{r}]+\sum_{i\in J_2}[\binom{a}{r}-\binom{a_i}{r}]\\
&\overset{(b)}{\leq}\sum_{i\in J_1}[\binom{a}{r}-\binom{a_i-1}{r}]+\sum_{i\in J_2}[\binom{a}{r}-\binom{a_i}{r}]\\
&\leq k_{max}(\sum_{i\in J_1}(a-a_i+1)+\sum_{i\in J_2}(a-a_i))
\end{aligned}$$

Note (a) follows from the fact that $\binom{x+m_1+1}{r}-\binom{x+1}{r}\geq\binom{x+m_1}{r}-\binom{x}{r}$ and (b) follows using similar reasoning.

Now

$$\begin{aligned}
&\sum_i a_i = t_1.a+(t-t_1).(a+1)\\
\overset{(a)}{\Longrightarrow}&\sum_{i\in I_1}a_i+\sum_{i\in I_2}a_i+\sum_{i\in J_1}a_i+\sum_{i\in J_2}a_i = a.(|I_2|+|J_2|)+(a+1).(|I_1|+|J_1|)\\
\Longrightarrow&\sum_{i\in I_2}(a_i-a)+\sum_{i\in I_1}(a_i-a-1)=\sum_{i\in J_2}(a-a_i)+\sum_{i\in J_1}(a+1-a_1)
\end{aligned}$$

Note that (a) follows from the fact that $|I_1\cup J_1\cup\{i\in[t]:a_i=a+1\}|=t-t_1$ and $|I_2\cup J_2\cup\{i\in[t]:a_i=a\}|=t_1$ Since we prove previously that $k_{min}\geq k_{max}$, we argue that

$$\begin{aligned}
&\sum_{i\in I_1}\Big[\binom{a_i}{r}-\binom{a+1}{r}\Big]+\sum_{i\in I_2}\Big[\binom{a_i}{r}-\binom{a}{r}\Big]\geq\sum_{i\in J_1}[\binom{a+1}{r}-\binom{a_i}{r}]+\sum_{i\in J_2}[\binom{a}{r}-\binom{a_i}{r}]\\
\Longrightarrow&\sum_{i\in I_1\cup I_2\cup J_1\cup J_2}\binom{a_i}{r}\geq|I_1+J_1|.\binom{a+1}{r}+|I_2+J_2|.\binom{a}{r}\\
\overset{(a)}{\Longrightarrow}&\sum_i\binom{a_i}{r}\geq t_1.\binom{a}{r}+(t-t_1).\binom{a+1}{r}
\end{aligned}$$

Note (a) follows from the fact that $|I_1\cup J_1\cup\{i\in[t]:a_i=a+1\}|=t-t_1$ and $|I_2\cup J_2\cup\{i\in[t]:a_i=a\}|=t_1$ and definitions of I and J.

□

Theorem 1. *If any scheme robust to s stragglers with n workers can recover l gradients with maximum load per worker denoted by r_{max} for gradient computation, then $\frac{\binom{s}{t}}{\binom{n}{t}}\leq 1-\alpha$ where $t=\lceil\frac{n.r_{max}}{k}\rceil$*

Proof. We know $\sum_i y_i\leq n.r_{max}$ where y_i is defined in the previous lemma.

Also, we know $\sum_{i=1}^{k}\binom{n-y_i}{n-s}\leq\binom{n}{s}k(1-\alpha)$

Now define $a=\lfloor\frac{\sum_{i=1}^{n}y_i}{k}\rfloor$ and $k_1=(a+1)n-\sum_{i=1}^{k}y_i$, thus from the previous claim,

$k_1\binom{n-a}{n-s}+(k-k_1)\binom{n-a-1}{n-s} < \sum_{i=1}^{k}\binom{n-y_i}{n-s}$. Clearly, the LHS is the smallest when $\sum_{i=1}^{k} y_i = n.r_{max}$ since a increases with increase in $\sum y_i$

Thus, the inequality becomes

$k\binom{n-a-1}{n-s} \leq \sum_{i=1}^{k}\binom{n-y_i}{n-s} \leq \binom{n}{s}k(1-\alpha)$ where $a = \lfloor \frac{n.r_{max}}{k} \rfloor$ since $\binom{n-a-1}{n-s} \leq \binom{n-a}{n-s}$ which proves the above theorem.

□

3 Gradient Coding Schemes for the modified problem

3.1 Scheme I:

Divide n workers in $(s+1)$ buckets of $(\frac{n}{s+1})$ workers each and cyclically allot α fraction of the data to each bucket. Within each bucket the data is disjointly and equally distributed among the workers of the bucket. So, even if 1 worker from each bucket straggles, the remaining one bucket will give us the partial gradient. The incidence matrix for this scheme will be as follows:

$$\begin{bmatrix} \underbrace{1....1}_{l} & 0 & 0 & .... & 0 & 0 \\ \underbrace{0....0}_{\text{n/(s+1)}} & \underbrace{1....1}_{l} & 0 & 0 & .... & 0 \\ \underbrace{0....0}_{\text{n/(s+1)}} & \underbrace{0....0}_{\text{n/(s+1)}} & \underbrace{1....1}_{l} & 0 & .... & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 1 & 1 & \cdots & 0 & 0 & \underbrace{1....1}_{\text{n/(s+1)}} \end{bmatrix}_{(s+1)\times n}$$

Note: Each row of this matrix corresponds to a bucket of workers, hence the matrix is $(s+1)\times n$ instead of $n \times n$.

$$Load/worker = \frac{l(s+1)}{n} \tag{11}$$

Note that each part of the data is given to $\frac{l(s+1)}{n}$ workers on average which is weaker than the bound we have in 22.

3.2 Scheme II:

$$\begin{bmatrix} \star & \star & .... & \star & 0 & 0 & ... & 0 & 0 \\ 0 & \star & \star & .... & \star & 0 & ... & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & .... & \star & \star & \star & \star \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ \star & .... & \star & \star & 0 & 0 & .... & 0 & \star \end{bmatrix}_{n\times n}$$

In this scheme,

$$Load/worker = s + l - n + 1 \tag{12}$$

Note that each part of the data is given to $s + l - n + 1$ workers on average which is weaker than the bound we have in 22.

One can observe that, the computation load per worker is lesser for scheme II than for scheme I. Scheme II only guarantees that every set of $(n - s)$ has access to α fraction of the dataset. The coeffecients in place of '$\star$' need to be chosen such that *Condition 2* should be satisfied. It turns out that for a general case it is not possible to find such coefficients. Hence, to use this scheme one will need to bear the excess communication costs that come along as we may have to send the entire set of gradients.

3.3 Scheme III:

In this scheme we make the following assumption. Each part as defined at the start of the problem can be further divided into many data-points. This may be valid for very high-dimensional datasets. Here we assume the entire dataset is divided into $N = \binom{n}{y}$ chunks and each chunk is treated separately. Recall that we want to compute the gradients of l parts, and the entire dataset is comprised of n parts.

Each chunk of data can be indexed using y distinct indices(i.e. a vector of y distinct indices). Note that for this scheme, $B \in \mathcal{R}^{n\times\binom{n}{y}}$. Any data chunk labelled by $\{a_1, a_2, a_3, \ldots\ldots, a_y\}$ is assigned to worker W_i if and only if $i = a_j$ for some $j \in [y]$.

However in this scheme we assume that each worker sends the entire set of gradients computed by it, not a linear combination of these. Thus, there is a high communication cost compared to other scheme I discussed.

$$Load/worker = n\frac{\binom{n-1}{y-1}}{\binom{n}{y}} = n\frac{y}{n} = y \tag{13}$$

Note that load denotes the number of parts computed by each worker.

The condition $\frac{\binom{s}{y}}{\binom{n}{y}} \leq 1-\alpha$ would ensure that α fraction of the dataset can be computed whenever any set of s workers straggle. The smallest y satisfying the above equation can be chosen and hence we can say that this scheme is the best one satisfying 22.

3.4 Scheme IV

The schemes discussed above are deterministic in the sense that α fraction of the gradient was recovered in each iteration with probability 1. Sections III and IV of [2] discuss a scheme to retrieve the partial(fractional) gradient with high probability. The construction of the BRC consists of two layers. In the first layer, the original data set $\{D_i\}_{i=1}^{n}$ are partitioned into n/b batches $\{B_i\}_{i=1}^{n/b}$ with the size of each batch equal to b. The intermediate coded partial gradients can be represented by $g_i^b = \sum\limits_{j=1+(i-1)b}^{ib} g_j$. In the second step, a type of raptor code is constructed taking the coded partial gradients g_b as input block.

Given the degree distribution $P \in \mathcal{R}^{n/b}$ and batches $\{B_i\}_{i=1}^{n/b}$ definition 5 of [2] defines the (b, P)-batch rapter code as: each worker $k \in [n]$, stores the data (B_i) and computes,

$$\tilde{g_k} = \sum_{i \in I} g_i^b = \sum_{i \in I} \sum_{j=1+(i-1)b}^{ib} g_j \tag{14}$$

where I is a randomly and uniformly subset of $[(\frac{n}{b})]$ with $|(I)| = d$, and d is generated according to distribution P.

Theorem 6 of [2] gives the degree distribution P required to successfully retrieve α fraction of the gradient and further states that this scheme achieves an *average computation load* of

$$O\Big(\frac{\log(\frac{n}{n-l})}{\log(\frac{n}{s})}\Big) \tag{15}$$

Note that, asymptotically this average load is the same as that we got in Eq.(11) or $Condition 3$. The downside of using batch raptor codes is that the load is not distributed uniformly among all the workers.

This can be seen as follows. As we know that asymptotically, $\binom{n}{y}$ can be written as n^y orderwise, condition 22 becomes $y > \frac{\log(\frac{n}{n-l})}{\log(\frac{n}{s})}$ thus the average load as described in Scheme III becomes $y = O(\frac{\log(\frac{n}{n-l})}{\log(\frac{n}{s})})$ if we choose the smallest y satisfying condition 3.

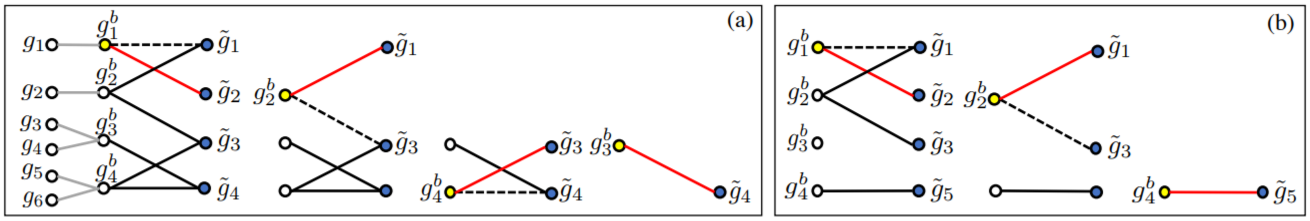


Figure 2: Decoding using peeling algorithm([2])

As shown in Fig. 2 a peeling algorithm is used for decoding. This relies on the fact that with high probability, a ripple(degree one node) exists at each peeling step. This is precisely the reason why a specific (b, P) need to be chosen.

References

[1] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding," 2016.

[2] S. Wang, J. Liu, and N. Shroff, "Fundamental limits of approximate gradient coding," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 3, no. 3, p. 1–22, Dec 2019. [Online]. Available: http://dx.doi.org/10.1145/3366700

4 A construction attempt for scheme II

We consider the case of $n = 6$, $s = 3$. Here let us choose $l = 5$. Let us follow the cyclic scheme where we distribute same number of parts to each worker.

Workers	a_1	a_2	a_3	a_4	a_5	a_6
b_1	$\times$	$\times$	$\times$			
b_2		$\times$	$\times$	$\times$		
b_3			$\times$	$\times$	$\times$	
b_4				$\times$	$\times$	$\times$
b_5	$\times$				$\times$	$\times$
b_6	$\times$	$\times$				$\times$

As per our construction, we want a vector $v \in \{0,1\}^n$ s.t $||v|| \geq 5$ chunks in the span of any $n-s$ workers. We know that if we distribute $s+1$ parts to each worker it would be the standard scheme as discussed in an earlier work where any set of $n-s$ workers can recover the entire $a_1+a_2+a_3+a_4+a_5+a_6$. Let us see if we can do something better as the master node is satisfied with $v \in \{0,1\}^n$ s.t $||v|| \geq 5$.

Let us distribute $s=3$ parts to each worker and each worker returns a linear combination.

Suppose b_1, b_3, b_4 be the only workers that return. Note that a_1 and a_2 are present in only b_1.

There are 3 cases possible:

Case - I: The computed vector at master does not contain a_1. In this the coefficient of a_1 in worker b_1 is zero. However in this case, if worker b_1,b_2 and b_3 return, the support at master won't contain a_1 at all, thus the vector can't have have a support of 5 elements.

Case - II: The computed vector at master does not contain a_2. In this the coefficient of a_2 in worker b_1 is zero. Thus worker b_1 returns only a linear combination of a_1 and a_3.

Case-III: The computer vector contains both a_1 and a_2. In this case, the coefficient of a_1 and a_2 in worker b_1 is same.

Hence only second and third case are possible.

Similarly, consider only worker b_1, b_4 and b_5 to be returning data to the master server.

Again using a similar logic we can show that worker b_1 must have either coefficient of a_2 to be zero or a_2 and a_3 have same coefficient.

Thus we can show that worker b_1 must either return $c_{1,1}.a_1 + c_{1,3}a_3$ or $d_1(a_1+a_2+a_3)$.

Now,we can argue this is the case for every worker in the list.

Now we show that if this is the case then there would exist a triplet which don't have the desired vector in span.

Case -I: There is only one vector which has a linear combination of just 2 parts. W.L.O.G we assume it to be b_1.

Now suppose b_2, b_3 and b_4 return. Since all have the same coefficients of their parts, we can argue that they can't return $a_2+a_3+a_4+a_5+a_6$.

Case-II: There are 2 workers which have a linear combination of 2 parts and they are consecutive.W.L.O.G we assume them to be b_1 and b_2.

Again if b_3, b_4 and b_5 return, the master can't compute $a_3+a_4+a_5+a_6+a_1$.

Case-III: There are exactly 2 workers which return a linear combination of 2 parts, the locations of which differ in their position by one. WLOG, we assume them to be b_1 and b_3.

Again if b_4,b_5 and b_6 return, the master can't compute $a_1+a_2+a_4+a_5+a_6$.

Case-IV: There is exactly 2 workers which return a linear combinations of two parts, the locations of which differ by 2. WLOG, we assume them to be b_1 and b_4.

Again if b_1, b_2 and b_3 return we can't compute $a_1+a_2+a_3+a_4+a_5$.

Case- V: There are exactly 3 workers which return a linear combination of two parts. Here we cannot compute the desired vector at master node if the other three workers return.

Case - VI: No worker transmits a linear combination of two parts. In this case each vector transmits the sum of 3 parts assigned to it. Clearly, this won't work if b_4,b_5 and b_6 straggle.

Thus, clearly transmitting one linear combination won't work in this case.

4.1 Generalisations

Actually, this argument can be generalised to show that under this scheme the coefficient of every part must be same in each worker i.e. each worker j returns $\sum_{i \in S} a_i$ if S denotes the set of parts present in worker b_j. We also assume that coefficient of every part given to a worker must be non-zero.

We argue that this need not always lead to a contradiction.

Consider this cyclic example where $n = 7$, $s = 3$ and $l = 6$. Thus in this cyclic scheme we assign $s + 1 + l - n = 3$ parts to each worker.

Workers	a_1	a_2	a_3	a_4	a_5	a_6	a_7
b_1	×	×	×				
b_2		×	×	×			
b_3			×	×	×		
b_4				×	×	×	
b_5					×	×	×
b_6	×					×	×
b_7	×	×					×

Like in this case, if every worker transmits the sum of its parts we can recover the one desired at master for example using just b_1 and b_4 we can compute $a_1 + a_2 + a_3 + a_4 + a_5 + a_6$.

In general, it can be shown that if l is a multiple of $s + 1 + l - n$ then such a scheme always works, else it doesn't.

5 A construction attempt for scheme III

We consider the case of $n = 5$, $s = 3$, $y = 2$ Let us try to tabulate the job distribution of each chunk to every worker. As per the construction we have $\binom{5}{2} = 10$ chunks

Workers	$a_1(1,2)$	$a_2(1,3)$	$a_3(1,4)$	$a_4(1,5)$	$a_5(2,3)$	$a_6(2,4)$	$a_7(2,5)$	$a_8(3,4)$	$a_9(3,5)$	$a_{10}(4,5)$
b_1	×	×	×	×						
b_2	×				×	×	×			
b_3		×			×			×	×	
b_4			×			×		×		×
b_5				×			×		×	×

As per our construction, we want a vector $v \in \{0, 1\}^n$ s.t $||v|| \geq 7$ chunks in the span of any $n - s$ workers. We attempt to construct a code such that each worker transmits exactly a linear combination of the gradients it is computing.

Suppose any three servers straggle, say only b_1 and b_2 reply. Since we need $a_1 + a_2 + a_3 + a_4 + a_5 + a_6 + a_7$ at the main server, the coefficients of a_5,a_6 and a_7 in worker b_2 has to be the same and the coefficients of a_2, a_3, a_4 in worker b_1 has to be the same.

Consider any general case that only workers b_i and b_j do not straggle. In that case all the chunks corresponding to the pairs (i,l), $l \neq j$, must have the same coefficient in worker b_i.

Now consider the case that only workers b_i and b_k do not straggle for some $k \neq j$. In this case, the chunk corresponding to the pairs (i,l), $l \neq k$ must have the same coefficient in worker b_i.

Combining the two results, we can say that all chunks must have the same coefficient in the linear combination calculated in worker b_i. Using a very similar line of argument, we can say that the coefficients of chunks must be the same for any other worker too in its linear combination calculated.

Now consider 2 workers b_1 and b_2. We know that chunks a_1, a_2, a_3 and a_4 in worker b_1 have the same coefficient and the chunks a_1, a_5, a_6 and a_7 have the same coefficient in worker b_2 as well. If this is the case we can't get $a_1 + a_2 + a_3 + a_4 + a_5 + a_6 + a_7$ in the span of the sums returned by workers b_1 and b_2.

So one attempt is to transmit more than one linear combination by each worker. We can actually show that it in this case if every worker transmits 3 linear combinations, then we can actually ensure the master server can compute what is desired (slight improvement over transmitting all 4 chunks).

So one question would be to ask whether it can be generalised and how many minimum linear combinations it to be transmitted by each worker. Clearly one combination won't work.

5.1 Construction using three linear independent combinations for each worker

Let us consider the workers b_1,b_2,b_3,b_4 and b_5. For every worker we look at the next 2 workers to the right of it. However, we consider a cyclic arrangement like b_1 is to right of b_5 and so all.

Intuition: Consider a pair of workers (i,j), $i \neq j$ returning jobs to the master. Suppose $i - j = 1$ or 2(note it is a cyclic subtraction like modulo arithmetic) the worker b_i contributes a vector with support of all the chunks computed by it while the worker b_j contributes a vector with 3 support chunks which were not present in worker b_i.

Exactly a similar thing happens if the reverse is true i.e $i - j = -1$ or -2.

So now we propose the construction.

Worker b_1 transmits $a_1 + a_2 + a_3 + a_4$, $a_2 + a_3 + a_4$ and $a_1 + a_3 + a_4$.

Worker b_2 transmits $a_1 + a_5 + a_6 + a_7$, $a_1 + a_6 + a_7$ and $a_1 + a_5 + a_7$.

Worker b_3 transmits $a_2 + a_5 + a_8 + a_9$, $a_2 + a_5 + a_9$ and $a_2 + a_5 + a_8$.

Worker b_4 transmits $a_3 + a_6 + a_8 + a_{10}$, $a_6 + a_8 + a_{10}$ and $a_3 + a_6 + a_8$.

Worker b_5 transmits $a_4 + a_7 + a_9 + a_{10}$, $a_7 + a_9 + a_{10}$ and $a_4 + a_9 + a_{10}$.

Thus for each worker b_i, we have removed only those two support elements which were present in b_{i+1} and b_{i+2}.

We can show that this construction would ensure the desired vector can be computed at master whenever any three straggle.

6 A general construction with slightly lower communication cost when n is coprime to y

Associate each data-part to any set of y workers of the n workers. Consider data-part i being denoted by the set $I_i \subseteq [n]$, $|I_i| = y$. Recall that data-part i is given to every worker in I_i. For each data-part i, choose a worker A_i s.t. $A_i \in I_i$ such that $|\{i : i \in [\binom{n}{y}] A_i = j\}| = 1/y \times \binom{n-1}{y-1}$ for every j which would ensure the constraint of balancing. We now design such a construction when n is co-prime to y.

Consider all possible integer solutions to the equations s.t $x_1 + x_2 + ... + x_y = n$ s.t $1 \geq x_i \geq n$ $\forall i \in [y]$. Let us denote a solution by $I_a = [a_1, a_2, ...a_y]$ Consider all possible y cyclic shifts of this solution i.e $[a_r, a_{1+r\%y}, ..., a_{1+(r+y-2)\%y}]$ for every $2 \leq r \leq y$. Now we create a list of all possible solutions to the above equation so that no solution in the list is a cyclic shift of any other solution of the above equation. Let us denote the entire set of such solutions where none is a cyclic shift of another by the set J.

Consider any element of J say $j \in J$. We construct a list of y elements with a as the first element such that the consecutive differences are given by the elements in list j i.e. the set of elements is constructed as $\{a, 1 + (a + a_1 - 1)\%n, 1 + (a + a_1 + a_2 - 1)\%n, ..., 1 + (\sum_{i=1}^{y-1} a_i - 1)\%n\}$. Note that by construction of J, no other element in J can create the same set with any integer else it would be a cyclic shift of j.

Now we prove that if n is co-prime with y, for every element $j \in J$, choosing a different starting element a creates a different set of y elements. Suppose not. We denote j by the list $[j_1, j_2, ..., j_y]$

Since the shifts are cyclic, W.L.O.G, we assume that stating element 1 and a give the same set of y elements and we choose the smallest such integer a. Suppose the $[1 + (r + t)\%y]^{th}$ element created in the list for starting element 1 be equal to the t^{th} element created in the list for starting element a $\forall t \in [y]$, thus $\sum_{z=1}^{r} j_z = a - 1$ and in general $\sum_{z=1}^{r} j_{1+(t+z-1)\%r} = a - 1$ for every integer t. i.e. any consecutive set of r elements element has the sum to be $a - 1$.

Suppose y is not a multiple of r. Suppose not and say the remainder when r divides y is given by q, then we can argue that $\sum_{z=1}^{q} j_{1+(t+q-1)\%r}$ remains the same for all t clearly the sum of which is smaller than $a - 1$, thus there would exist an integer smaller than a say b such that b and 1 give the same set of y elements.

Thus y is a multiple of r say $y = r.m$, hence $n = \sum_{z=1}^{y} j_z = m. \sum_{z=1}^{r} j_z = m.(a - 1)$ since any the sum of any set of r consecutive elements remain the same. Thus n and y have the same factor implying they are not-coprime.

Hence for every element $j \in J$, choosing a different starting element gives a different set of y elements.

Now we define the following function f from all possible set of y elements to the set of integers in $[n]$.

Consider any list in J and create a set of y elements using this list as defined above using starting element a for every a and the function f would map this set to a. Since we have previously shown that all sets created using different lists in J would be unique, we can argue that each element in $[n]$ would have the same number of pre-images in its domain under function f.

Now for each data part i define A_i=$f(I_i)$. Since every element in range of f has the same number of pre-images in its domain, the condition $|\{i : i \in [\binom{n}{y}] A_i = j\}| = 1/y \times \binom{n-1}{y-1}$ is enforced.

Consider the matrix A of size $n \times \binom{n}{y}$ which denotes the assignments of data-parts to different workers. However if data-part i is not present in worker j, put X at that position. Else put 1 in that position.

Clearly, as described in scheme III each part occurs y times. Now, consider all sets of positions in which any part i occurs. For each data-part i, we exactly replace the coefficient for worker A_i from 1 to 0.

Now each worker transmits the sum of the gradients parts assigned to it along with the gradients of those parts which are assigned 1 in the matrix A. Thus the communication load for each worker can be given by $1 + \lceil \frac{(y-1).\binom{n-1}{y-1}}{y} \rceil$, thus the communication load reduces approximately by a factor of $\frac{y-1}{y}$.

6.1 Reconstruction from the transmission

Consider any set of any $n - s$ workers denoted by I. We denote the set of parts which are assigned to exactly 1 worker in I by $W_{1,I}$. We denote the set of workers where part i is assigned to by P_i. We denote the set of parts which are spanned by the workers in set I by W_I i.e the smallest set W_I satisfying $i \in W_I \ \forall i \in I$.

Choose the smallest set of workers M such that $i \in M \ \forall i \in W_{1,I}$. For each worker in I, consider a transmission which is the sum of all the gradients of the parts assigned to it. Now sum all these parts up. Clearly, the coefficient of each part in $W_{1,I}$ in the above sum would have its coefficient as 1. Now consider the parts which occurred more than once in the sum. Each such part must be present in more than one worker in I, thus at least one worker in I would be directly transmitting that part as per the scheme designed above and thus subtract it with appropriate multiplier from the obtained sum calculated above and we thus recover the sum of the parts denoted by W_I.

We can show that under these communication, the master can recover the desired sum of the gradients.

7 Proof of the cyclic coding scheme described above

Consider the cyclic coding scheme described in scheme II where each worker is assigned a sum of $s + 1 + l - n$ parts consecutively.

$$\begin{bmatrix}
\underbrace{[1....1]}_{r} & \underbrace{[1....1]}_{r} & ...\underbrace{[1....1]}_{r} & \underbrace{1\ 1....1}_{n-l} \\
1\underbrace{[1....1]}_{r} & \underbrace{[1....1]}_{r} & ...\underbrace{[1....1]}_{r} & \underbrace{1....1}_{n-l-1} \\
11\underbrace{[1....1]}_{r} & \underbrace{[1....1]}_{r} & ...\underbrace{[1....1]}_{r} & \underbrace{1....1}_{n-l-2} \\
\vdots & \vdots & \ddots & \ddots \\
\underbrace{1\ldots1}_{r-1} & \underbrace{[1....1]}_{r} & \underbrace{[1....1]}_{r} & \underbrace{1....1}_{n-l-r+1}
\end{bmatrix}$$

Though from the diagram it is not so clear the end 1 of the first rectangle in the second row is just below the first 1 of the second rectangle in the first row and this holds true in general. Also note that we have just shown l rectangles i.e. l workers in the matrix above.

Note that in each row we have exactly $\frac{l}{r}$ such [] where each bracket denotes a worker with the support elements in the positions contained in []. Each of these r rows would span a vector of support weight l.

Consider any set of $r - 1 + n - l$ stragglers. Clearly if each row does not have one worker corresponding to each rectangle straggled then we would have a linear vector of support length l in the span.

Thus, we assume at least one rectangle in each row is straggled.

Consider the rightmost straggled worker in any row and consider a preceding straggled worker in the same row. If between the two workers we have another straggling worker in another row, take the rightmost such worker and repeat the same step. If not, then repeat the same process on the preceding worker in the same row. Repeat the process till we have no straggled worker in the same row.

Now we consider this straggling worker and shift the rectangle corresponding to it and the workers right of it in the same row by probably different number of steps so that none of these rectangles clash with a straggled worker.

Consider this straggled worker and suppose we reached this worker after visiting $x+1$ different rows, thus there would be at least x straggling workers behind it.

Thus to ensure that no number of shifts can ensure a disjoint set of rectangles we need to have at least $n-l$ straggling workers ahead of the worker considered finally.

Equivalently this last statement can be written as the minimum number of integer values no $\{x_i\}_{i\in[m]}$ can take such that $0 < x_m \leq n-l+(m-1).r$ given $x_{i+1}-x_i \geq r$ has no integer solution which can be shown to be $n-l$. Note that m would denote the number of workers present in the row ahead of the straggled worker (including the straggled worker)where the above algorithm terminated.

Also given that we have visited $x+1$ different row shifts, we haven't visited at least $r-x-1$ rows each of which must also have a straggled worker. Thus the minimum of stragglers needed would be $x+(r-x-1)+(1)+(n-l) = r+n-l$ (1 is added to consider the straggled worker where the algorithm terminated) to ensure that there exists no vector of support length l in the span, however since we have $r-1+n-l$ stragglers, we can have the vector of ones and zeros with exactly l ones in the span.

7.1 A formal structure to this proof

Recall that worker W_i contains the data subsets $D_i, D_{i+1}, ..D_{1+(i+r-2)\%r}$ where $r = s+1+\beta-n$. Let the workers denoted by $W_1, ..., W_\beta$ be grouped into r groups each group containing $\frac{\beta}{r}$ workers. Let us denote the groups by $\{A_j\}_{j\in[r]}$. Suppose group A_j contains the $\frac{\beta}{r}$ workers $W_j, W_{j+r}, ...W_{j+\beta-r}$ which ensures that no one worker is present in two different groups. Note that worker denoted by W_i belongs in group $A_{f(i)}$ where $f(i) = 1+(i-1)\%r$.

Consider the set of straggling workers be denoted by S s.t $|S| = s$. The set of straggling workers among the first β workers is denoted by T.

Algorithm 1: Stopping Straggler

```
Choose largest $i \in [\beta]$ s.t $i \in S$.
**while** $\exists\ j < i\ s.t\ W_j \in A_{f(i)}\ and\ W_j \in [S]$ **do**
    Choose largest $j < i$ s.t $W_j \in A_{f(i)}$.
    **if** $\exists k\ s.t\ j < k < i\ and\ W_k \in [S]$ **then**
        Choose largest $k$ s.t $j < k < i$ and $W_k \in [S]$.
        i =k.
    **else**
        i =j
    **end**
**end**
Output $i$.
```

Note that worker W_i returned would have no worker W_j in $A_{f(i)}$ s.t $j < i$ and $W_j \in S$.

Now consider the entire set of groups visited in the algorithm described as I. We can argue if $I = x + 1$, there must exist at least x workers W_j satisfying $j < i$ and $W_j \in S$ (at least one worker from each of the x rows). Also note that there must be at least one straggling worker corresponding to each of $(r - x - 1)$ groups which were not visited in the algorithm. Note that each of these workers must have its index smaller than i, thus there would exist at least $(r - x - 1) + x = r - 1$ workers behind W_i.

Now consider worker W_i in group $A_{f(i)}$ and suppose we have m workers W_j s.t. $j \geq i$ and $W_j \in A_{f(i)}$. Suppose we denote all the workers in group $A_{f(i)}$ as $W_{i_1}, ..., W_{i_{\frac{\beta}{r}}}$, thus $i = i_{\frac{\beta}{r} - m + 1} = f(i) + (\frac{\beta}{r} - m)r$

We now claim there must exist at least a vector $[k_1, k_2, ..., k_m]$ satisfying

$$k_t - k_{t-1} \geq r, \ \ , k_1 \geq i \tag{16}$$

$$k_m \leq n + f(i) - r \tag{17}$$

$$W_{k_t} \notin S \ \forall t \in [m] \tag{18}$$

Note that the conditions mentioned above would ensure that no overlap between the data subsets assigned to workers $W_{i_1}, W_{i_2}, ..., W_{i_{\frac{\beta}{r} - m}}, W_{k_1}, ..., W_{k_m}$.

Since $i = f(i) + (\frac{\beta}{r} - m)r$, it we may equivalently calculate the minimum size of set Z s.t there is no solution in non-negative $\{k_v\}_{v \in [m]}$ to (16) can be given by $(n + f(i) - r) - i - (m-1).r + 1 = n + f(i) - m \times r - i + 1 = n + f(i) - m \times r - f(i) + (\frac{\beta}{r} - m)r + 1 = n - \beta + 1$

However, the number of straggled workers W_j s.t $j \geq i$ is at most $s - r + 1 = n - \beta$ which would imply that there exists at least a vector $[k_1, ..., k_m]$ satisfying (16), thus proving the existence of a set of $\frac{\beta}{r}$ non-straggling workers such that the data subsets assigned to them don't overlap.

7.2 A formal structure to proof when $s + 1 + \beta - n$ does not divide n

Recall that worker W_i contains the data subsets $D_i, D_{i+1}, ..D_{1+(i+r-2)\%r}$ where $r = s + 1 + \beta - n$. Note that we denote the remainder by x when r divides n. Let the workers denoted by $W_1, ..., W_\gamma$ be grouped into r groups each group containing $\frac{\beta}{r}$ workers where $\gamma = \beta - x$ which clearly divides r. Let us denote the groups by $\{A_j\}_{j \in [r]}$. Suppose group A_j contains the $\frac{\gamma}{r}$ workers $W_j, W_{j+r}, ...W_{j+\gamma-r}$ which ensures that no one worker is present in two different groups. Note that worker denoted by W_i belongs in group $A_{f(i)}$ where $f(i) = 1 + (i-1)\%r$.

Consider the set of straggling workers be denoted by S s.t $|S| = s$.

Note that worker W_i returned from the Algorithm 2 would have no worker W_j in $A_{f(i)}$ s.t $j < i$ and $W_j \in S$.

Now consider the entire set of groups visited in the algorithm described as I. We can argue if $I = z + 1$, there must exist at least z workers W_j satisfying $j < i$ and $W_j \in S$ (at least one worker from each of the x rows). Suppose there exists m workers W_j s.t $j \geq i$ and $W_j \in A_{f(i)}$ and we denote all the workers in group $A_{f(i)}$ as $W_{i_1}, ..., W_{i_{\frac{\gamma}{r}}}$, thus $i = i_{\frac{\gamma}{r} - m + 1} = f(i) + (\frac{\gamma}{r} - m)r$

Case I: There does not exist any group without straggler.

Thus there exists exactly $(r - z - 1)$ groups which are not visited each of which must have a straggling worker with index smaller than i. Thus there exist at least $(r - z - 1) + z = r - 1$ workers with index smaller than i.

Now consider the smallest index u larger than j such that W_u is a non-straggling worker.

Algorithm 2: Stopping Straggler

```
Choose largest $i \in [\gamma]$ s.t $i \in S$.
while $\exists\ j < i\ s.t\ W_j \in A_{f(i)}\ and\ W_j \in [S]$ do
    Choose largest $j < i$ s.t. $W_j \in A_{f(i)}$.
    if $\exists k\ s.t\ j < k < i\ and\ W_k \in [S]$ then
        Choose largest $k$ s.t $j < k < i$ and $W_k \in [S]$.
        i =k.
    else
        i =j
    end
end
Output $i$.
```

We now claim there must exist at least a vector $[k_1, k_2, ..., k_m]$ satisfying

$$k_t - k_{t-1} \geq r, \ \ , k_1 \geq u + x \tag{19}$$

$$k_m \leq n + f(i) - r \tag{20}$$

$$W_{k_t} \notin S\ \forall t \in [m] \tag{21}$$

Note that the conditions mentioned above would ensure that no overlap between the sum of all the r data subsets transmitted by the workers $W_{i_1}, W_{i_2}, ..., W_{i_{\frac{\gamma}{r}-m}}, W_{k_1}, ..., W_{k_m}$ and the data subset of first x gradients transmitted by the worker W_u. Recall from the definition of γ that $\gamma = \beta - x$.

The minimum size of set Z s.t there is no solution of $\{k_v\}_{v\in[m]}$ to $k_v \notin Z \forall v \in [m]$ and (19) can be shown to be $n + f(i) - r - (u + x) - (m-1)r + 1 = n + f(i) - (u + x) - m \times r + 1 = (n-u) + (i-\beta) + (f(i) - i + \gamma) - m \times r + 1 = (n-u) + (i-\beta) + 1$.

However note that since W_u is the smallest index non-straggling worker larger than i, there are at least $u - i$ straggling workers from W_i to W_u. However, since the total number of stragglers starting from W_{u+x} is $(s - (r-1) - (u-i) = -\beta + n - u + i$ which is clearly smaller than the minimum size needed to ensure no solution of (19).

Thus we can recover the desired sum of β gradients from the transmission of sum of r gradients by the workers $W_{i_1}, W_{i_2}, ..., W_{i_{\frac{\gamma}{r}-m}}, W_{k_1}, ..., W_{k_m}$ and the sum of x gradients by worker W_u

Case -II: There exist groups $\{A_j\}$ with no straggling workers in any of these groups.

Case (a): The largest and the smallest indices of such groups without any straggling worker differ by at least x. Suppose the largest index of such groups is denoted by L_1 and the smallest index of such group is denoted by L_2. Suppose the workers of group A_{L_1} is denoted by $\{W_{T_1}, W_{T_2}, ..., W_{T_{\frac{\gamma}{r}}}\}$. Note that $T_1 = L_1$ and $T_i - T_{i-1} = r$. We can show that the data subsets of workers $W_{T_1}, W_{T_2}, ..., W_{T_{\frac{\gamma}{r}}}$ and the first x data subsets of W_{L_2} are non-overlapping since $L_1 - L_2 \geq x$, thus the master can recover the sum of β gradients at the master.

Case (b): The largest and the smallest indices of such groups without any straggling worker differ by a quantity smaller than x. Suppose the set of all such indices is denoted by J with $|J| = t$. Since it is a cyclic scheme and cyclic shifts in data subsets assigned to workers don't make any difference, we assume the smallest and largest index of J to be 1 and a where $a < x + 1$.

Now consider $B_w = A_w \cup \{W_{w+(\gamma)}\} \forall w \in J$ and $w \geq r - x$ Note that since the largest and the smallest index of J differ by a quantity smaller than x, there can be any common worker between two distinct subsets B_i and B_j $i, j \in J$. Also note that $w + \gamma \leq n$ since $w \leq x$ and $n \geq \beta$.

Suppose there exists $w \in J$ s.t no worker in B_w straggles, then choose the master can recover the sum of β gradients by the sum of r gradients of workers A_w and the sum of first x gradients transmitted by $W_{w+\gamma}$.

Suppose there does not exist any worker $w \in J$ s.t no worker in B_w straggles. In this case, each worker in $W_{w+(\gamma)} \forall w \in J$ must straggle as the other workers don't straggle. Choose the largest element in J which has been assumed to be a in this case and thus consider the worker $W_{a+\gamma}$. The worker with index i has clearly smaller index than all workers in $W_{w+(\gamma)} \forall w \in J$. Also we know there are at least $(r - t - 1)$ straggled workers with indices smaller than i. Thus we have $(r - t - 1) + 1 + (t - 1) = r - 1$ straggled workers with indices smaller than $a + \gamma$.

We denote the workers in group A_a as $\{W_{a_1}, W_{a_2}, ..., W_{a_{\frac{\gamma}{r}}}\}$.

$$z - a_{\frac{\gamma}{r}} \geq r, \ , a_{\frac{\gamma}{r}} = a + \gamma - r \tag{22}$$

$$z \leq n + a - x \tag{23}$$

$$W_z \notin S \tag{24}$$

Note that the conditions mentioned above (if satisfied) would ensure that no overlap between the sum of all the r data subsets transmitted by the workers $W_{a_1}, W_{a_2}, ..., W_{a_{\frac{\gamma}{r}}}$ and the data subset of first x gradients transmitted by the worker W_z. Recall from the definition of γ that $\gamma = \beta - x$.

The minimum size of set Z s.t there is no solution of z to $z \notin Z \forall v \in [m]$ and (22) can be shown to be $n + a - x - (a + \gamma - r) - r + 1 = n - \beta + 1$.

However, the number of straggling workers with indices larger than or equal to $a_{\frac{\gamma}{r}}$ is $(s - r + 1) = (n - \beta)$ which is clearly smaller than the minimum size of Z needed to ensure no solution in (22). Thus we can recover the desired sum of β gradients from the transmission of sum of r gradients by the workers $W_{a_1}, W_{a_2}, ..., W_{a_{\frac{\gamma}{r}}}$ and the sum of x gradients by worker W_z.

8 Lower bounds under certain constraints

Problem setting: We have a set of n parts to n workers, we wish to distribute some parts to each worker and each worker transmits a linear combination to the master. We wish to ensure that even if any s workers straggle, the master should be able to compute some vector $v \in \{0, 1\}^{\text{n}}$, with $||v|| \geqslant l$.

In the cyclic coding scheme, we had $r = s + 1 + l - n$ parts to each worker and were able to prove that we can't recover the desired vector if every worker just transmits one vector provided $r > 1$ and l is not a multiple of r.

Now we try to prove a lower bound on maximum load r_{max} on any worker. Clearly, if $s + 1 + l - n = 1$, then the cyclic scheme would work as every worker has just one part.

Case- I: $s + 1 + l - n = 1$: In this case the cyclic scheme itself gives the lower bound, thus $r_{max} = 1$

Case-II: $s + 1 + l - n = 2$. We can show that $r_{max} > 2$ if l is odd else $r_{max} = 2$. We can argue that coefficients of different parts in each worker has to be the same if we assign two parts to each worker. We then construct a graph with nodes as each part and an edge exists between two nodes if there exists a worker with both of these parts together. Since there are n nodes and n edges, this graph can be a connected one with just one cycle or the graph may have multiple disconnected components with each component having a cycle if we assume that each part occurs at least once in some worker. Straggling s workers is like removing s edges and we can show that each part would have the same coefficient in the linear combination transmitted by each worker. We can extend it to the case where some part does not occur at all as well.

However, this does not imply that there does not exist any construction. If we can ensure that l is a multiple of 2, then we can actually ensure the desired result like a cyclic construction discussed in scheme 2.

However, if l is odd and $s = 2$, we can show that there would exist a cut of 2 edges which would ensure that a desired vector of length l can't be returned. I believe this result can be extended beyond $s = 2$ as well as long as l is odd and $s + 1 + l - n = 2$ which if true would imply $r_{max} > 2$

The following theorem gives a lower bound on the number of data chunks assigned to each worker if each worker transmits just one linear combination of the gradients computed by it.

Theorem 2. *Consider a dataset divided into n uniform chunks and the master needs the sum of the gradients corresponding to any l chunks. Suppose we have n workers with each worker having a maximum load of r. Then, if l is odd and the master has to compute the desired vector under any $s = n - l + 1 \geq 2$ stragglers, we show that $r > 2$.*

Note the linear vector transmitted by any worker does not have zero coefficient for any of the parts present in it. If it is so, we can equivalently assume that this part was not present in the worker at all.

Proof. Let us prove by contradiction. Suppose we have $r = 2$ i.e. every worker transmits and computes a linear combination of gradients of at most two data-parts.

We represent every data-parts as a node in the graph. Since every worker can be assigned at most 2 data-parts, we denote it as an edge if it indeed transmits a linear combination of two data-parts, else we represent it a self-loop around a node corresponding to the data-part assigned to it.

Case-(a): We assume a uniform distribution of two data-parts to every worker with each data-part being assigned to at exactly two workers.

Since every gradient is being computed by two workers, each node must be present in two edges. Also note that we have n edges and n nodes, thus the graph must be comprised of disjoint cyclic components with each component being a cycle. There could be a pair of isolated nodes with a pair of edges connecting them which we treat as a connected cyclic component only like the component C in the figure3.

Let us denote the sizes(vertics/edges) of the components by $c_1, c_2, ...c_t$ if there are t such components with $\sum c_t = n$ W.L.OG, we assume $c_1 \leq c_2 \leq c_3 ... \leq c_t$. Let p denote the smallest index such that $\sum_{i=1}^{p} c_i \geq s + 1$.

Note that the lines on edges denoting the straggling workers in the diagram.

We first prove that the coefficient of each gradient transmitted by each worker is the same.

Suppose not and there exists an worker where the components are not the same.

Case-I: Such an edge exists in some component with size larger than or equal to $s + 1$. For simplicity, we denote the size of this component by x and the node of this component is $A_1, A_2, ..., A_x$ and the edges are $A_1 - A_2, A_2 - A_3, ...A_x - A_1$.

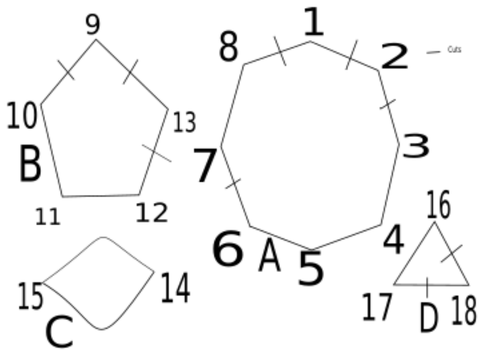


Figure 3: Representation as parts as nodes with numbers denoting nodes and letters denoting components

We assume the edge corresponding to this worker as $A_1 - A_2$. Now we consider the case when workers corresponding to edges $A_x - A_1$, $A_2 - A_3$,$A_3 - A_4$,...,$A_s - A_{s+1}$ straggle- for example the cut in component B. In this case clearly the gradients corresponding to parts $A_3, .., A_s$ cannot be computed at all, thus these $s - 2$ gradients cannot be present in the sum computed by the master. However the master can have at most $n - l = s - 1$ missing gradients and the gradients corresponding to the parts $A_1 - A_2$ is present in only worker would imply that both coefficients of the gradients in the worker must be same.

Case-II: Such an edge exists in some component r of size smaller than or equal to s.

For simplicity, we denote the size of this component by x and the node of this component is $A_1, A_2, ..., A_x$ and the edges are $A_1 - A_2, A_2 - A_3, ...A_x - A_1$.

Let p_{min} be the minimum index of i such that $\sum_{j=1;j\neq r}^{i} c_i \geq s - x + 1$ and $p_{last} = s - x + 1 - \sum_{j=1;j\neq r}^{p_{min}-1} c_i$.

Thus, we straggle the following s workers-

- All workers in the first $p_{min} - 1$ components excluding component numbered r.
- Exactly p_{last} continuous edges of component c_i like the cut shown in component B.
- All edges except $A_1 - A_2$ in component numbered r example - component D in the diagram.

Now we can clearly see that $s - x + x - 2 = s - 2$ gradients won't be available at the master if the above set of s workers straggle. However, the master can have at most $n - l = s - 1$ missing gradients which would imply at least gradient A_1 or gradient A_2 must be computed by the master. Also note that parts A_1 and A_2 is present in just one worker and the master must have coefficient of all the gradient to be the same, which would further imply that the coefficient of parts A_1 and A_2 in the worker corresponding to edge $A_1 - A_2$ must be the same.

Thus, we prove that the coefficients of each part in each worker must be the same. Note that upto this part, no where did we assume that l is odd.

However, we show that if l is odd, this construction cannot yield the desired gradient at the master.

We know that the sizes of the components are $c_1, c_2, ..., c_t$ with $\sum_t c_t = n$ and choose $d_1, d_2, ..., d_n$ such that $\sum_t d_t = s$ with $d_i = c_i \forall 1 \leq i < m$ for some m, $d_m \leq c_m$ and $d_i = 0 \forall i \geq m + 1$.

The selection of stragglers show that at least $s-1$ gradients cannot be computed at the master. None of the gradients corresponding to any of parts present in any of the first $m-1$ workers can be computed by the master. Also $c_m - d_m$ must clearly be even as $\sum_t (c_t - d_t)$ is even.

Suppose $c_m - d_m$ is odd, then there there must exist some $i \geq m+1$ s.t c_i is odd as $\sum_{i \geq m+1}(c_i - d_i) = l - 1 - c_m - d_m$ which is odd. Thus, we assume c_j to be odd for some $j \geq m+1$. Now we decrease d_m by 1 and set d_j to 1. Note that this ensures that $\sum d_t$ remains s.

Now let us define the s workers which straggle. All the workers corresponding to the edges in the first $m-1$ components straggle and a set of continuous d_m edges in component numbered m straggle and continuous d_j edges in component numbered j straggle.

Consider component numbered m. The workers which don't straggle correspond to a set of continuous $c_m - d_m$ edges which is even and hence only a sum of $c_m - d_m$ gradients could be obtained corresponding to that cycle.

Similarly in component numbered j, there are exactly $c_j - d_j$ workers corresponding to a set of continuous edges which don't straggle. Either the workers which don't straggle form the entire set of edges in the cycle in which case we obtain the sum of gradients corresponding to all the parts in the cycle, or the workers which don't straggle form a continuous set of even edges potentially excluding one edge in the cycle in which case only a sum of $c_t - d_t$ gradients corresponding to the parts in the cycle can be computed.

For all other cycles where no worker is straggled we can obtain the sum of gradients of all the parts in the cycle i.e. a sum of gradients of $c_i - d_i = c_i$ parts.

Thus, we could obtain a sum of gradients of exactly $\sum_{i=m+1}^{t} c_i - d_i = n - s = l - 1$ parts which contradicts the requirement that the master should compute a sum of gradients of at least l parts.

Case-(b): Suppose the above graph is composed of t distinct components with no component having more than one cycle in it.

Suppose we have exactly y workers which are assigned exactly one gradient of which exactly z of them have been assigned data-parts which are assigned to some other worker assigned two data-parts, thus $y - z$ workers have been assigned single data-parts for computation which is not present in any edge. Suppose these $y - z$ workers span g distinct edges. As there could be multiple workers assigned a single gradient for computation, $g \leq y - z$.

Since g nodes are not covered by any edge coupled with the fact that each component can have at most one cycle, we argue that there would be exactly g components without any cycle. Suppose the maximum number of edges (self-loop or connecting two nodes) that can be removed without reducing the number of cycles in the graph is denoted by c_{max}. Clearly if we start straggling these edges in order from an edge starting from a leaf node, each removal of an edge causes at least a node getting removed from the span of the remaining edges.

Thus, if the total number of straggling workers is less than the maximum number of edges that can be removed without affecting any cycle, we can argue that no sum of more than $n - s = l - 1$ gradients can be computed from the non-straggling workers.

Now suppose the number of straggling workers is larger than or equal to $c_{max} + 2$, we can show that every worker corresponding to every edge in every cycle would have the same coefficients of the gradients of the data-parts transmitted by it. We can argue this by straggling workers in exactly the same way as described in the previous case ass after straggling c_{max} workers we have only cycles remaining similar to previous case and at least 2 more workers that have to be straggled.

However, if the number of straggling workers is $c_{max} + 1$, we may have to straggle workers in a slight different way.

Suppose there are more than 2 vertices in a cycle into which paths from leave nodes branch into. Well in this case, actually we can show that all workers corresponding to every edges in the cycle would have the same coefficient for both the parts. Suppose A_1, A_2, A_3 be three consecutive nodes in the cycle and from each node there exists a cyclic path to a leaf node. Now we straggle the entire path to a leaf node from A_2, the edge $A_p - A_1$ (assuming p nodes in the cycle) and edge $A_2 - A_3$, the entire path from a leaf node to A_3 except the edge which connects it to the cycle. Using these set of stragglers we can show that the worker corresponding to the edge $A_1 - A_2$ has same coefficient for both the data-parts. Similarly, we can argue that the workers corresponding to all edges have the same coefficient for both the workers and a similar selection of stragglers will show that no sum of l gradients exists in the linear span of non-straggling workers.

Suppose there at most two vertices in the cycle into which paths from leaf nodes merge into. In this case, there might be at most two edges in the cycle the workers corresponding to which may not have the same coefficient for the data-parts assigned to them.

Suppose there exists only a path to node A_1 in the cycle to a leaf node of length c_{max}, thus in this example we can show that all workers except those corresponding to edges $A_1 - A_2$ and $A_n - A_1$ have the same coefficient for both the gradients of the data-parts assigned to it.

Now if p(the edges in cycle) is odd, straggle the workers corresponding to the edges in the path from A_1 to leaf node and the edge $A_1 - A_2$. We can show that since an even number of consecutive edges remain in the component after straggling the edges we cannot have a sum of all the gradients of the data-parts contained in the span of existing edges. Since l data-parts remain in the span after the selection of straggling workers, we cannot have any sum of l gradients in its span.

Suppose p is even, straggle the workers corresponding to the edges in the path from A_1 to leaf node and the edge $A_1 - A_2$. Now the number of edges remaining in this component is odd which would imply that there must exist some other non-straggling cyclic component too as number of non-straggling edges which is $n - s = l - 1$ is even. Also there must exist a cyclic component after removal of c_{max} edges which has an odd number of edges. Thus instead of straggling node $A_1 - A_2$, straggle a node in another cyclic component with odd number of edges and the same argument as above follows.

Case (c): Suppose we remove all relaxations and there exist components with multiple cycles and some components with no cycles at all.

Now consider the maximum number of edges (or self-loops) (denoted by set E_1) that can be removed so that no cycle in any component is removed. After these edges consider the maximum number of edges that can be removed (denoted by set E_2) so that there exists at most one cycle in every component which had a cycle.

Note like the previous case order the edges in E_1 have been ordered along a path starting from any leaf node and the edges in E_2 should also be ordered along a path and a new path should only be chosen only when no more edges can be removed without breaking another cycle.

Now remove the edges in order from E_1 and E_2 in $|E_1| + |E_2|$ steps and note after any t steps in the process, we have at most $n - t$ nodes in the span of remaining edges. Thus, if $s \leq |E_1| + |E_2|$, we are able to get a contradiction that there would exist a set of s straggling workers so that there exists no sum of $l = n - s + 1$ gradients in the span of remaining non-straggling workers.

If $s > |E_1| + |E_2|$, we can choose stragglers in a very similar way as described in case (b) and show a contradiction in a very similar way. This is because after the removal of $|E_1| + |E_2|$ workers, only cycles remain in the graph and the situation becomes quite similar to that described in case (b).

Corollary 1. *Consider a dataset divided into n uniform chunks and the master needs the sum of the gradients corresponding to any l chunks. Suppose we have n workers with each worker having a maximum load of r. Thus if the master has to compute the desired vector under any $s = n-l+1 \geq 2$ stragglers, we show that $r > 2$.*

Proof. Let us prove by contradiction. Suppose we have $r = 2$ i.e. every worker transmits and computes a linear combination of gradients of at most two data-parts.

We represent every data-parts as a node in the graph. Since every worker can be assigned at most 2 data-parts, we denote it as an edge if it indeed transmits a linear combination of two data-parts, else we represent it a self-loop around a node corresponding to the data-part assigned to it.

Suppose the graph is represented in t distinct components and suppose u components have no cycle in it. There might be some components with multiple cycles in it too.

Now we define an order of straggling workers. First start with those components which don't have any cycle and straggle the workers corresponding to those edges which start from a leaf edge and straggle workers continuously along a path. After all the workers corresponding to edges in non-cyclic components are straggled, straggle those edges which are not part of any cycle again starting from a leaf edge and straggle continuously along a path. Now choose a component with more than one cycle and straggle workers continuously along a path after one cycle is reduced. Workers are straggled along a new path only when no more edges can be removed without a breaking a new cycle. Note that this process would ensure a connected component remains connected even while straggling workers are removed as edges or self-loops from the graph.

At the end, we would have only cyclic components remaining. Straggle the edges in each cycle continuously till no edge in a cycle is left and then start with the next cycle. We stop when there is no worker is left.

Note that in this process of straggling, when $n - s$ workers are remaining we can have atmost $n - s + 1$ vertices or data partions being spanned. This can be argued from the fact that unless we break a new cycle a data partition is always removed from the span whenever a worker is straggled.

Thus straggling of s workers under the above mentioned process would ensure at most $n-s+1 \leq \beta - 1$ data partitions being available at the master.

□

□

9 Construction of coding schemes for the case l is odd when $s + 1 + l - n = 2$

We showed in the previous example that under $s + 1 + l - n = 2$, if l is odd, load per worker of 2 won't work. When l is even, we showed that the cyclic scheme clearly works.

When l is odd, we consider 2 cases namely $n - l$ is even and $n - l$ is odd.

Case - I: $n - l$ is even.

Since $n - l$ is even, n is odd. Let us label the data parts as $a_0, a_2, ..., a_{n-1}$. Since n is odd we can write $n = 2\alpha + 1$ for some integer α.

For the first $\alpha + 1$ workers we assign three parts to each worker. For each $0 \leq i \leq \alpha$, the parts a_{2i}, $a_{(2i+1)\%n}$ and $a_{2i+2\%n}$ is assigned to worker b_i. For $\alpha + 1 \leq i \leq 2\alpha$, the parts $a_{2.(i-\alpha-1)+1}$

and $a_{2.(i-\alpha-1)+2}$ are assigned to it. Each worker transmits the sum of the gradients of the parts assigned to it.

Consider the following example with $n = 9, l = 5$ and $s = 5$

Workers	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
b_0	×	×	×						
b_1			×	×	×				
b_2					×	×	×		
b_3							×	×	×
b_4	×	×							×
b_5		×	×						
b_6				×	×				
b_7						×	×		
b_8								×	×

We can indeed show that it indeed satisfies the constraint when any set of s workers straggle.

Case - II: $n - l$ is odd.

Since $n - l$ is odd, n is even. Let us label the data parts as $a_0, a_2, ..., a_{n-1}$. Since n is even we can write $n = 2\alpha$ for some integer α.

For the first $\alpha + 1$ workers we assign three parts to each worker. For each $0 \leq i \leq \alpha - 1$, the parts a_{2i}, $a_{(2i+1)\%n}$ and $a_{2i+2\%n}$ is assigned to worker b_i. We assign the parts a_{n-1},a_1 and a_2. For $\alpha+1 \leq i \leq 2\alpha-1$, the parts $a_{2.(i-\alpha-1)+1}$ and $a_{2.(i-\alpha-1)+2}$ are assigned to it. Each worker transmits the sum of the gradients of the parts assigned to it.

Consider the following example with $n = 8, l = 7$ and $s = 2$

Workers	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
b_0	×	×	×					
b_1			×	×	×			
b_2					×	×	×	
b_3	×						×	×
b_4	×	×						×
b_5		×	×					
b_6				×	×			
b_7						×	×	

We can indeed show that it indeed satisfies the constraint when any set of s workers straggle.

10 Cyclic coding schemes with communication cost 2 when $s+1+l-n$ does not divide l

Now, consider a scenario where l is not a multiple of $r = s + 1 + l - n$ and suppose the remainder when l is divided by r is denoted by x.

Consider the cyclic scheme as described in the previous section and each worker transmits the summation of all the parts assigned to it and the summation of the first x parts assigned to it. Note this cyclic scheme has the same number of parts to every worker i.e. $s + 1 + l - n$ and we can show using a similar method that it is tolerant to s stragglers.

11 Generalisation of the construction described above

In this scheme let us consider the example where we give r parts and $r+1$ parts to each worker and suppose $l = a.r + b.(r+1)$ where $a < r+1$. Let $n - l = c.r + d$ where $d < r$.

So under this scheme, we design the cyclic coding scheme but with some modifications. Consider the first $ar+1$ workers. Each of these workers is allotted consecutively r gradients cyclically. However from $(ar+2)^{th}$ worker, we allot each worker exactly $(r+1)$ parts. Suppose the worker $(ar+1)$ had gradients from part t to $t+r-1$, then the worker $(ar+2)$ also has gradients from part t to $t+r$, post this worker the assignment is purely cyclic.

Let us consider $n = 16$ and $l = 10$

Workers	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
b_0	$\times$	$\times$	$\times$													
b_1		$\times$	$\times$	$\times$												
b_2			$\times$	$\times$	$\times$											
b_3				$\times$	$\times$	$\times$										
b_4					$\times$	$\times$	$\times$									
b_5						$\times$	$\times$	$\times$								
b_6							$\times$	$\times$	$\times$							
b_7							$\times$	$\times$	$\times$	$\times$						
b_8								$\times$	$\times$	$\times$	$\times$					
b_9									$\times$	$\times$	$\times$	$\times$				
b_{10}										$\times$	$\times$	$\times$	$\times$			
b_{11}											$\times$	$\times$	$\times$	$\times$		
b_{12}												$\times$	$\times$	$\times$	$\times$	
b_{13}													$\times$	$\times$	$\times$	$\times$
b_{14}	$\times$													$\times$	$\times$	$\times$
b_{15}	$\times$	$\times$													$\times$	$\times$

Under this type of construction, we can show that this construction is tolerant to $s = -\min(a,c) + n - l + r$ stragglers. Since a can take the minimum value of $r+1$, this can take a minimum of $l - n - 1$.

Though this type of construction does not attain the straggler tolerance we can show that if $n - l$ differs from $s + 1 + l - n$ by a constant factor β, there would exist a construction with $s + 1 + l - n + \beta + 1$ parts to every worker

12 Some other alternate constructions:

Suppose $l = a.r + b.(r+x)$ for some integer r and $n - l = c(r+x) + d$ where $d \leq x$. Under these constraints we can design a scheme which is tolerant to $r - 1 + c.r + d$ stragglers.

So under this scheme, we design the cyclic coding scheme but with some modifications. Consider the first $ar + x$ workers. Each of these workers is allotted consecutively r gradients cyclically. However from $(ar+x+1)^{th}$ worker, we allot each worker exactly $(r+x)$ parts. Suppose the worker $(ar+x)$ had gradients from part t to $t+r-1$, then the worker $(ar+x+1)$ also has gradients from part $t-x+1$ to $t+r$, post this worker the assignment is purely cyclic.

Consider $l = 11$ where $r = 3$ and thus $11 = 3.2 + 5$ and $n = 13$ thus $n - l = 2$

Workers	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}
b_0	×	×	×										
b_1		×	×	×									
b_2			×	×	×								
b_3				×	×	×							
b_4					×	×	×						
b_5						×	×	×					
b_6							×	×	×				
b_7								×	×	×			
b_8							×	×	×	×	×		
b_9								×	×	×	×	×	
b_{10}									×	×	×	×	×
b_{11}	×									×	×	×	×
b_{12}	×	×									×	×	×

This scheme is tolerant to $r - 1 + n - l = 2 + 2 = 4$ stragglers.

13 Lower bound attempts under constraint $n - l > \frac{s}{2}$

We consider balanced assignments in this consideration i.e each worker has the same number of parts and each part is given to the same number of workers. Suppose the load peer worker is denoted by r, thus each part is present in r workers.

Now, consider a pair of parts. Consider the number of workers in which either of two is present. Clearly, the number of workers in which either is present must by less than or equal to $2r$. If $r \leq s + 1 + l - n$ and $n - l > s/2$, $2r < s + 2$, thus number of workers in which either one of the two is present can be at most $s + 1$, implying that the coefficients associated with both the parts in all the corresponding must be the same. Arguing similarly for every pair of parts, we can say that each worker must have the same coefficient for all the parts implying the coefficient associated for each part w.r.t every worker is 1.

Now let us consider a balanced load r per worker as defined above and coefficient for all parts in each worker to be the same.

We believe that if $r < s + 1 + l - n$, there won't exist a sum of l parts as its linear combination for any set of s stragglers.

We assume l to be a multiple of r and denote l/r to be x. If it is not the multiple, e may have to consider the ceiling function.

Each worker has r parts with it. Consider all possible disjoint sets of x workers (workers whose parts don't overlap) and denote each such collection by S_i. Suppose there are t such collections. Clearly no two sets are exactly identical. Also denote the set of such collections in which worker i is present by A_i.

Using a union-intersection property of sets, we know for $L \in [n]$ with $|L| = s$;

$$| \cup_{u \in L} A_u| = \sum_i |A_i| - \sum_{i<j} |A_i \cap A_j| + \sum_{i<j<k} |A_i \cap A_j \cap A_k| - ..$$

Also we can show that $\sum_i |A_i| = t.x$ as each collection has l workers and $\sum_{i<j} |A_i \cap A_j| = t.\binom{x}{2}$. Similarly, we can show $\sum_{i_1<i_2<..<i_t} |A_{i_1} \cap A_{i_2} ... \cap A_{i_m}| = t.\binom{x}{m}$

Thus, taking summation over all such terms we can say that

$$\sum_{L\in[n]:|L|=s} |\cup_{u\in L} A_u| = \binom{n-1}{s-1}t.x - \binom{n-2}{s-2}t\binom{x}{2} + .. + (-1)^{r+1}\binom{n-s}{0}t\binom{x}{s} = t\Big[\binom{n}{s} - \binom{n-x}{s-1}\Big]$$

The last term follows by choosing two polynomials such that the above summation become the coefficient for a particular power of the their product. Also not that there are $\binom{n}{s}$ such collections in $[n]$ and the above summation also decreases with r and thus we may believe there would exist r_0, such that if $r < r_0$, there would exist a list L of cardinality s such that $|\cup_{u\in L} A_u| = t$ implying if the stragglers are chosen from the set L, no summation of any l parts would lie in its span.

Currently not exactly sure how to find this r_0, though I feel r_0 could possibly be close the load under the cyclic scheme $s + 1 + l - n$.