Combinatorical analysis of burst failures for large-scale cluster

Meng Wang

November 22, 2022

1 Setup

We consider a storage system of N drives such that $N = X \cdot Y \cdot Z$, where there are X racks in the system, each rack contains Y enclosures, and each enclosure contains Z drives.

Let $M = Y \cdot Z$, so M denotes the number of drives per rack.

In particular, we are considering ORNL Alpine system, which is composed of 39 racks. 38 racks have 8 enclosures each, and 1 rack has 4 enclosures. Each enclosure has 106 drives.

For simplicity, we assume the system contains 40 racks. Each rack contains 8 enclosures. Each enclosure contains 100 drives.

2 Total instances with fixed number of affected racks

Consider f failures happen in r racks.

We first choose r racks from all the X racks, which has C_X^r combinations.

Given r racks $1, 2, ..., r$, denote $T(f, r)$ as the total number of instances for f failures to happen in r racks, such that each rack has at least one failure.

Consider the r-th rack. If it has i failures, then the rest $r - 1$ racks must have $(f - i)$ failures in total with each rack having at least one failure.

Therefore, we have the following recurrence relation:

$$T(f, r) = \sum_{\substack{1 \leq i \leq f \\ i \leq M}} C_M^i \cdot T(f - i, r - 1) \tag{1}$$

The base case is

$$T(f, 1) = \begin{cases} C_M^f & \text{if } 1 \leq f \leq M \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Therefore, we can compute $T(f, r)$ using dynamic programming, with time complexity $O(f \cdot r)$, and memory complexity $O(f \cdot r)$. An implementation can be found at: `https://github.com/ucare-uchicago/mlec-sim/blob/main/src/theory/policies/total.py`

The total number of instances is:

$$C_X^r \cdot T(f, r) \tag{3}$$

3 Survival instances under local clustered (RAID)

Consider $k_l + p_l$ local-only RAID SLEC. For easier deployment we assume $n_l = k_l + p_l$ is divisible by Z.

n_l drives in the same enclosure compose a RAID disk group. Therefore a rack contains $g = M/n_l$ RAID groups.

3.1 Survival counting in a single RAID group

Denote $\Gamma(f, g)$ as the number of instances for one single rack to survive f failures in a rack containing g $k_l + p_l$ RAID groups.

For $\Gamma(f, g)$, we have the following recurrence relation (which is derived by considering what will happen if g-th group contains i failures):

$$\Gamma(f, g) = \sum_{\substack{0 \le i \le p_l \\ a \le f}} \Gamma(f - i, g - 1) \cdot C_{n_l}^{i} \tag{4}$$

The base case is

$$\Gamma(f, 1) = \begin{cases} C_{n_l}^{f} & \text{if } 0 \le f \le n_l \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

We can then compute $\Gamma(f_i, g_l)$ based on recurrence relation 4 and dynamic programming, with $O(f \cdot g)$ time complexity and $O(f \cdot g)$ memory complexity.

3.2 Survival counting in entire system

Given r racks $1, 2, ..., r$, denote $S(f, r)$ as the total number of instances for $k_l + p_l$ local-only RAID to survive f failures to in r racks, such that each rack has at least one failure.

For $S(f, r)$, we have the following recurrence relation (which is derived by considering what will happen if r-th rack contains i failures):

$$S(f, r) = \sum_{\substack{1 \le i \le f \\ i \le M}} \Gamma(i, g) \cdot S(f - i, r - 1) \tag{6}$$

The base case is

$$S(f, 1) = \begin{cases} \Gamma(f, g) & \text{if } 1 \le f \le M \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Since we have already computed $\Gamma(f, g)$, we can further compute $S(f, r)$ based on recurrence relation 6 and dynamic programming. The total time complexity $O(f \cdot (g + r)$, and total memory complexity and $O(f \cdot (g + r)$.

Here is an example implementation: https://github.com/ucare-uchicago/mlec-sim/blob/main/src/theory/policies/raid.py

Therefore, the total number of survival instances in the whole system is:

$$C_X^r \cdot S(f, r) \tag{8}$$

Therefore, the probability of data loss under f failures on r racks for RAID is:

$$\text{RAID data loss} = \frac{C_X^r \cdot S(f, r)}{C_X^r \cdot T(f, r)} = \frac{S(f, r)}{T(f, r)} \tag{9}$$

4 Survival instances under local declustered parity

It's similar to local clustered erasure in Section 3, but now the size of the disk group is usually larger than n_l.

Suppose the size of the disk group is D, usually $n_l \le D \le Z$, where Z is the size of the enclosure.

If any disk group has more than p_l disk failures, then there is data loss.

So this time a rack contains $g = M/D$ disk groups.

We can then compute the data loss in the same way that we did for RAID. Therefore, the probability of data loss under f failures on r racks for local-only Declustered erasure is:

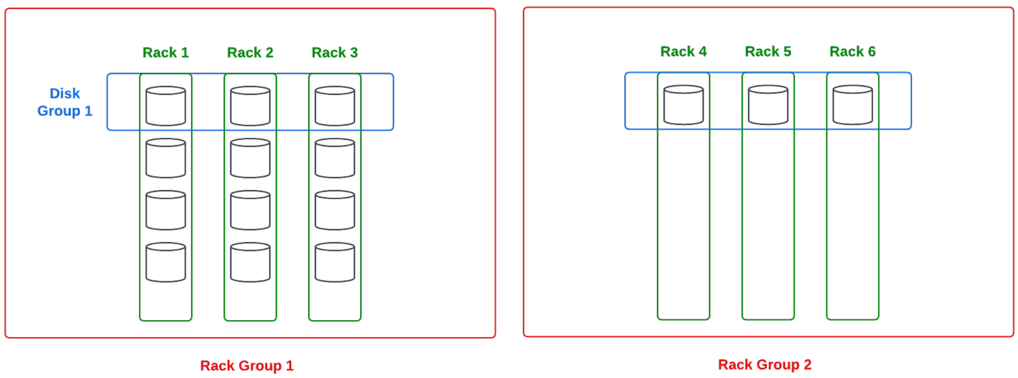
$$\text{Declustered data loss} = \frac{C_X^r \cdot S(f,r)}{C_X^r \cdot T(f,r)} = \frac{S(f,r)}{T(f,r)} \tag{10}$$

5 Survival instances under network clustered erasure

Consider $n_n = k_n + p_n$ network-only clustered erasure.

We first divide all the racks into $\alpha = X/n_n$ rack groups, each rack group with $n_n = k_n + p_n$ racks.

In each rack group, we further divide the disks into disk groups. Each disk group contains $n_n = k_n + p_n$ disks spread on n_n racks, with each disk on a different rack.



5.1 Survival counting in a single rack group

Denote $\Gamma(f,r,m)$ as the number of instances for one rack group to survive f failures in r racks in this rack group, with each rack contains m disks.

Thus there are m disk groups in the rack group: $1, 2, ..., m$. Suppose the m-th disk group has i failures spread in i racks. Then disk group $1, 2, ..., m-1$ should have $f-i$ failures in total. Suppose these $f-i$ failures spread in j racks. Then the i racks affected by disk group m and the j racks affected by disk groups $1...m-1$ must overlap on $i+j-r$ racks. And there are $r-j$ racks which are affected by disk group m only.

If we consider of the possible values of i and j, we can derive the following recurrence relation for $\Gamma(f,r,m)$:

$$\Gamma(f,r,m) = \sum_{\substack{0\le i\le p_n \\ i\le r}} \sum_{\substack{0\le j\le r \\ r-i\le j}} C_j^{i+j-r} \cdot C_{n_n-j}^{r-j} \cdot \Gamma(f-i,j,m-1) \tag{11}$$

The base case is

$$\Gamma(f,1,m) = \begin{cases} C_m^f \cdot C_{n_n}^1 & \text{if } 0 \le f \le m \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

We can then compute $\Gamma(f,r,M)$ based on recurrence relation 11 and dynamic programming, with $O(f \cdot r \cdot M)$ time complexity and $O(f \cdot r \cdot M)$ memory complexity.

5.2 Survival counting in the entire system

Given α rack groups $1, 2, ..., \alpha$, denote $S(f, r, \alpha)$ as the total number of instances for $n_n = k_n + p_n$ network-only RAID to survive f failures in r racks, such that each rack has at least one failure.

For $S(f, r, \alpha)$, we have the following recurrence relation (which is derived by considering what will happen if α-th rack group contains i failures affecting j racks):

$$S(f, r, \alpha) = \sum_{\substack{0 \le i \le f \\ i \le M * p_n}} \sum_{\substack{0 \le j \le r \\ j \le i \\ j \le n_n}} \Gamma(i, j, M) \cdot S(f - i, r - j, \alpha - 1) \tag{13}$$

The base case is

$$S(f, r, 1) = \Gamma(f, r, M) \tag{14}$$

Since we have already computed $\Gamma(f, r, m)$, we can further compute $S(f, r, \alpha)$ based on recurrence relation 13 and dynamic programming. The total time complexity $O(f \cdot r \cdot \alpha)$, and total memory complexity and $O(f \cdot r \cdot \alpha)$.

An example implementation can be found at: `https://github.com/ucare-uchicago/mlec-sim/blob/main/src/theory/policies/netraid.py`

6 Survival instances under network declustered erasure

This is easy.

Consider $n_n = k_n + p_n$ network-only declustered erasure.

There is data loss whenever there are more than p_n affected racks.

Therefore:

$$\text{Net-declus Data loss} = \begin{cases} 1 & \text{if } r > p_n \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

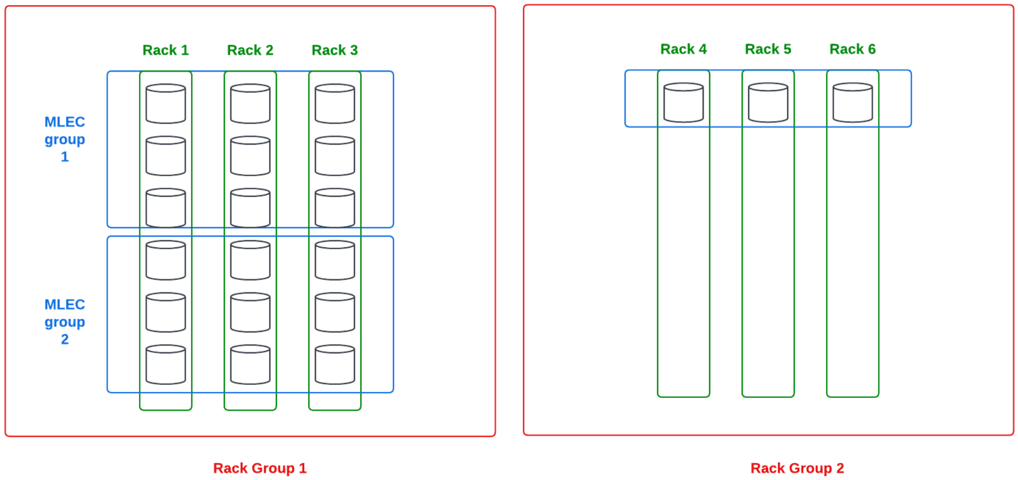
where r is the number of affected racks.

7 Survival instances under MLEC clustered

Consider $(k_n + p_n)/(k_l + p_l)$ clustered MLEC. We first divide all the racks into $\alpha = X/n_n$ rack groups, each rack group with $n_n = k_n + p_n$ racks.

In each rack group, we further divide the disks into mlec groups. Each mlec group contains $n_n = k_n + p_n$ disks on a different rack, thus $n_n \cdot n_l$ disks in total.

The following figure shows $(2 + 1)/(2 + 1)$ clustered MLEC layout among 6 racks.



7.1 Count survivals in a single mlec group

Consider a disk group spread among a racks, with each rack containing $n_l = k_l + p_l$ disks. Each rack can survive at most p_l disk failures. The whole disk group can survive at most p_n rack failures.

Denote $\Theta(a, p_n, f, r)$ as the number of instances for such a disk group to survive f failures in r racks in this rack group.

Assume the a-th rack contains i disk failures. If we consider of the possible values of i, we can derive the following recurrence relation for $\Theta(a, p_n, f, r)$:

$$\Theta(a, p_n, f, r) = \sum_{\substack{0 \le i \le n_l \\ i \le f}} \Delta(a, p_n, f, r, i) \tag{16}$$

where:

$$\Delta(a, p_n, f, r, i) = \begin{cases} \Theta(a-1, p_n, f, r) & \text{if } i = 0 \\ C_{n_l}^i \cdot \Theta(a-1, p_n, f-i, r-1) & \text{if } i \le p_l \\ C_{n_l}^i \cdot \Theta(a-1, p_n-1, f-i, r-1) & \text{if } i > p_l \end{cases} \tag{17}$$

The base case is

$$\Theta(a, p_n, f, 0) = \begin{cases} 1 & \text{if } f = 0 \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

Then the survival counting for a mlec group is $\Theta(n_n, p_n, f, r)$.

With dynamic programming, the time complexity and space complexity to compute $\Theta(a, p_n, f, r)$ are both $O(n_n \cdot p_n \cdot f \cdot r)$

7.2 Count survivals in a single rack group

Denote $\Gamma(f, r, m)$ as the number of instances for one rack group to survive f failures in r racks in this rack group.

The rack group contains $m = M/n_l$ mlec groups. Suppose the m-th rack group has i failures spread in j racks. Then mlec group $1, 2, ..., m-1$ should have $f - i$ failures in total. Suppose these $f - i$ failures spread in h racks. Then the j racks affected by disk group m and the h racks affected by disk groups $1...m-1$ must overlap on $j + h - r$ racks. And there are $r - h$ racks which are affected by disk group m only.

If we consider of the possible values of i,j, and h, we can derive the following recurrence relation for $\Gamma(f, r, m)$:

$$\Gamma(f,r,m) = \sum_{0 \le i \le f} \sum_{0 \le j \le r} \sum_{\substack{0 \le h \le r \\ r-j \le h}} C_j^{j+h-r} \cdot C_{n_n-h}^{r-h} \cdot \Theta(j, p_n, i, j) \cdot \Gamma(f-i, h, m-1) \tag{19}$$

The base case is

$$\Gamma(0,h,m) = \begin{cases} 1 & \text{if } h = 0 \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

With dynamic programming, the time complexity and space complexity to compute $\Gamma(f, r, m)$ are both $O(f \cdot r \cdot m)$

7.3 Survival counting in the entire system

Given α rack groups $1, 2, ..., \alpha$, denote $S(f, r, \alpha)$ as the total number of survivals under f failures in r affected racks.

For $S(f, r, \alpha)$, we have the following recurrence relation (which is derived by considering what will happen if α-th rack group contains i failures affecting j racks):

$$S(f,r,\alpha) = \sum_{\substack{0 \le i \le f \\ i \le M * n_n}} \sum_{\substack{0 \le j \le r \\ j \le i \\ j \le n_n}} \Gamma(i, j, M/n_l) \cdot S(f-i, r-j, \alpha-1) \tag{21}$$

The base case is

$$S(f,r,1) = \Gamma(f, r, M/n_l) \tag{22}$$

Since we have already computed $\Gamma(f, r, m)$, we can further compute $S(f, r, \alpha)$ based on recurrence relation 21 and dynamic programming. The total time complexity $O(f \cdot r \cdot \alpha)$, and total memory complexity and $O(f \cdot r \cdot \alpha)$.

The aggregate time complexity is $O(f \cdot r \cdot (n_n p_n + \frac{M}{n_l} + \frac{X}{n_n}))$

An example implementation can be found at: https://github.com/ucare-uchicago/mlec-sim/blob/main/src/theory/policies/mlec.py

8 Survival instances under MLEC Declustered

TBD