# Combinatorical analysis of burst failures for large-scale cluster

Meng Wang

November 17, 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:

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

The base case is

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

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/burst_theory.py#L30

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.

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

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

The base case is

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

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

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):

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

The base case is

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

Since we have already computed $\eta(f, g)$, 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/burst_theory.py#L97

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

TBD, this is more challenging.

## 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:

$$\text{Net-declus Data loss} = \begin{cases} 1 & \text{if } r > p_n \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

where $r$ is the number of affected racks.

## 7 Survival instances under MLEC clustered

TBD

## 8 Survival instances under MLEC Declustered

TBD