1 Introduction

Consider a tuple of containers of positive integral size $C = (c_1, c_2, \ldots, c_n)$, with $c_i \geq 0$, and a process which repeatedly reduces the size of some selection of s of those containers by one. We denote such a problem with the tuple (C, n, s). Let t be the number of steps taken before no set of containers of size s can be found with strictly non-zero remaining capacities. We wish to develop an upper bound, $t_{\max}$ for t, and show that the algorithm in section 3 achieves that bound.

We denote the sizes of containers at the "current" step as c_i, and at the following step as c_i'. Values following an arbitrary step x are denoted $c_i^{(x)}$.

2 Upper bound on the number of steps

A trivial upper bound, t_0, on $t_{\max}$ can be found by observing that

$$\sum_{i=1}^{n} c_i' = \sum_{i=1}^{s} (c_i - 1) + \sum_{i=s+1}^{n} c_i \tag{1}$$

$$= \left(\sum_{i=1}^{n} c_i \right) - s, \tag{2}$$

so each step reduces the available total capacity by s and therefore

$$t_0 = \left\lfloor \frac{\sum c_i}{s} \right\rfloor \tag{3}$$

is a bound on $t_{\max}$.

Now consider a partition of the problem into two parts, L and R, such that

$$|L| < s, \tag{4}$$

and

$$\min_{i \in L} c_i \geq \left\lfloor \frac{\sum_{i \in R} c_i}{s - |L|} \right\rfloor. \tag{5}$$

Observe that we can reduce the set of partitions which must be examined in this manner: If a partition exists which fulfils inequality 5, where $c_j \geq \min_{i \in L} c_i$ for some $j \in R$, we can move j from R into L without affecting the inequality, because the left-hand side is unchanged, and the right-hand side is made smaller. This means that we only need consider partitions where all containers smaller than some bound are placed in R, and those larger than or equal to the bound are placed in L, since we can always convert a bounding partition not of that form into a bounding partition of that form, without losing the bounding property.

With this in mind, we define the predicate $\mathbb{B}_q^{(x)}$ as

$$\mathbb{B}_q^{(x)} : c_q^{(x)} \geq \left\lfloor \frac{\sum_{i=q+1}^{n} c_i^{(x)}}{s - q} \right\rfloor, \tag{6}$$

where

$$c_1 \geq c_2 \geq \ldots \geq c_n \geq 0. \tag{7}$$

Theorem 1 *If $\mathbb{B}_q^{(x)}$ is true for some q, then*

$$t_q = \left\lfloor \frac{\sum_{i=q+1}^{n} c_j}{s - q} \right\rfloor \tag{8}$$

is an upper bound on the number of steps possible.

Proof: ...[1] $\square$

Corollary 1 *An upper bound on the number of steps possible is*

$$t_{\max} = \min\left(t_0, \min_{q:\mathbb{B}_q} t_q\right) = \min\left(\left\lfloor \frac{\sum_{i=1}^{n} c_i}{s} \right\rfloor, \min_{q:\mathbb{B}_q} \left\lfloor \frac{\sum_{i=q+1}^{n} c_j}{s-q} \right\rfloor\right). \tag{9}$$

3 Algorithm

The algorithm starts with a sorted sequence of containers, $c_1, c_2, \ldots, c_n$, meeting the ordering constraint (7).

We define a *step* in the algorithm, $C' = f_s(C)$ for some $s \leq n$, to be the process where the s largest containers are reduced in size by one, and the tuple re-ordered (with a stable sort, WLOG) to fulfil the constraint (7). The algorithm terminates when no more steps can be performed — i.e. when there are fewer than s bins with any free space in them. Specifically, the algorithm terminates when there is some c_i ($i \leq s$) where $c_i = 0$. This implies that $c_s = 0$ as well, by the ordering condition (7).

Theorem 2 *The state of $\mathbb{B}_q^{(x)}$ is preserved through a step of the algorithm: if $\mathbb{B}_q^{(x)}$ is true, then $\mathbb{B}_q^{(x+1)}$ is also true; if $\mathbb{B}_q^{(x)}$ is false, then $\mathbb{B}_q^{(x+1)}$ is also false.*

Proof: If $\mathbb{B}_q^{(x)}$ is true, then

$$c_q^{(x+1)} = c_q^{(x)} - 1,$$

since c_{i+1}^{x} can be at most ...

If $\mathbb{B}_q^{(x)}$ is false, then ...[2] $\square$

Since the state of $\mathbb{B}_q^{(x)}$ is preserved for all x, we may simply refer to $\mathbb{B}_q$.

4 Achievable bounds on the number of steps

We now prove that the bound from theorem 1 is achievable using the algorithm in section 3. We do this in three stages: first, we demonstrate that any problem may be decomposed into two parts, one with a trivial solution, and a reduced problem where $\mathbb{B}_q$ is false for all $1 \leq q < s$. We then show that if the reduced problem achieves its bounds, then the whole problem does. Finally we demonstrate that the reduced problem does achieve its bounds.

Consider a problem, (C, n, s). Now, there is either some q in equation 1 which gives the value to $t_{\max}$ (and we call that value of q the *bounding container*), or there is no such q and the trivial bound dominates.

Theorem 3 *For a problem (C, s, n) with no bounding container, the algorithm achieves the bound of $\left\lfloor \frac{\sum_{i=1}^{n} c_i}{s} \right\rfloor$ steps.*

Proof: If we cannot achieve the bound, we must terminate after some number of steps x, short of that bound. Specifically, we have the termination condition that $c_s^{(x)} = 0$, and more generally, $c_i^{(x)} > c_{i+1}^{(x)} = 0$ for some $i < s$. Now, since there is no bounding container, $\mathbb{B}_q$ is false for all $1 \leq q < s$, and in particular

[1] If such a partition exists, we can maximise the space used by reducing, at each step, every container in (the far larger) L, and as few in R as we can. Doing so does not change the relation (5), since it reduces $\min_{i \in L} c_i$ by at most 1, and $\sum_{i \in R} c_i$ by $s - |L|$, which is greater than one, from equation (4). In this process, we can place at most $\min_{i \in L} c_i$ steps by considering the set of containers L (since every container in L is reduced by 1 each step, we are bound by the minimum sized container in L). However, considering the set of containers R, we have a total capacity of $\sum_{i \in R} c_i$, and each step removes one unit of capacity from each of $s - |L|$ containers. Thus, we can place no more than

$$t_{\max} \leq t_{LR} = \left\lfloor \frac{\sum_{i \in R} c_i}{s - |L|} \right\rfloor$$

steps. Since this latter bound is smaller than the former, by (5), it dominates.

[2] This is the 3 diagrams with A and B cases

$\mathbb{B}_i$ is false. So, we have:

$$\begin{aligned}
0 &< c_i^{(x)} && \text{(termination condition)} && (10)\\
&< \left\lfloor \frac{\sum_{j=i+1}^{n} c_j^{(x)}}{s-i} \right\rfloor && (\mathbb{B}_i \text{ is false}) && (11)\\
&= 0, && (c_j^{(x)} = 0 \forall j > i) && (12)
\end{aligned}$$

which is a contradiction. □

Theorem 4 *In a problem, (C, n, s), with bounding container q, no container is moved at any step to the opposite side of q.*

Proof: □

Theorem 5 *In a problem, (C, n, s), with bounding container q, we may decompose it into two subproblems: $(L = \{c_1, \ldots, c_q\}, q, q)$ and $(R = \{c_{q+1}, \ldots, c_n\}, s-q, n-q)$, which are between them equivalent to the original problem, and that $t_{\max}(R) = t_{\max}(C)$.*

Proof: By theorem 4, no container is moved across the q boundary. This implies that the decomposed problems (L, q, q) and $(R, s-q, n-q)$ are always independent of each other. We can observe trivially that $t_{\max}(L) = c_q$, since every element of L is reduced by one at every step, and the algorithm must therefore stop when its smallest element, $c_q^{(x)} = 0$. Similarly, we can observe that $t_{\max}(R) \leq t_0(R) = \left\lfloor \sum_{i=q+1}^{n} c_i/(s-q) \right\rfloor$. Since q is the bounding container, $\mathbb{B}_q$ is true, so $t_{\max}(L) = c_q \geq t_0(R) \geq t_{\max}(R)$, and therefore the bound on R is the tighter bound, and so[3]

$$t_{\max}(C) = \left\lfloor \frac{\sum_{i=q+1}^{n}}{s-q} \right\rfloor = t_{\max}(R)$$

□

Theorem 6 *In a decomposition of the form in theorem 5, the problem $(R, s-q, n-q)$ has no bounding container.*

Proof:

□

Theorem 7 *The algorithm always achieves the upper bound (9), and is therefore optimal.*

Proof: If the problem has no bounding container, then by theorem 3, it achieves the trivial t_0 upper bound. Alternatively, if the problem has a bounding container q, then, by theorem 5, we may decompose it into an unconstrained part, (L, q, q), and a constrained part, $(R, s-q, n-q)$. The constrained part, by theorem 6, has no bounding container and therefore, by theorem 3, achieves its bound of

$$\left\lfloor \frac{\sum_{i=q+1}^{n} c_i}{s-q} \right\rfloor$$

steps. This is equal to the upper bound of the full problem, (C, s, n), since q is its bounding container. □

5 Conclusion

We have demonstrated that, for a fixed stripe width, the chunk allocator algorithm implemented in btrfs is optimal, and we can place a hard upper bound (equation 9) on the number of allocation steps that the algorithm can perform, for any given set of storage devices.

[3] does this follow?