AFS Server Balancing by Linear Programming

Russ Allbery

May 13, 2023

Abstract

The meaning of "balance" for AFS servers is discussed in mathematical terms. A phrasing of the AFS server balancing problem as a linear programming problem is presented and parameters for optimization are discussed. A trick for recovering the balancing solution in a form suitable for application is also shown.

Terminology

An AFS file server consists of one or more partitions of disk space, each of which holds one or more AFS volumes. An AFS volume is an indivisible unit of AFS storage with an associated current size, maximum quota, and access count. (For our purposes, we will define access count to be the number of acceses, read or write, to that volume in the past day; more accurate and less variable measurements can be made.) Each AFS volume must be located on one and only one server partition.

Let S be the set of volume locations that we intend to balance. This may either be a set of servers or a set of server partitions, depending on the desired granularity of the balance. Let V be the set of volumes that must be stored in those locations. We will use s to refer to an element of S and v to refer to an element of V.

Define functions $U(v)$ and $A(v)$ that, for a given volume v, return the current size of that volume and the access count for that volume respectively. This data can be obtained by querying the volserver for information about the volumes in the affected set.

Define a function $C(s)$ that, for a given location s, returns the capacity of that location (the total volume size that can be placed in that location).

Finally, define a boolean location matrix L such that $L[s, v] = 1$ if and only if the volume v is in location s before the balancing process. The goal of balancing is thus to produce a new boolean location matrix L' such that the arrangement of volumes is balanced.

Defining Balanced

A set of AFS server partitions is considered "balanced" when the total used disk space and total accesses are spread evenly across those server partitions so that each AFS server shares the load in proportion to its ability to bear the load. The purpose of this section is to develop a suitably rigerous definition of "evenly."

We want to divide number of volumes, total disk space, and number of accesses across the available locations in proportion to their capacity. This makes the assumption that locations with

a higher capacity can also support more accesses, a possibly dubious assumption but one that we have found holds reasonably well in practice. To that end, define a proportion function $P(s)$ as follows:

$$P(s) = \frac{C(s)}{\sum_{x \in S} C(x)}$$

We now define a location matrix L to represent a balanced distribution of volumes if, for each location s, the following constraints hold:

$$\begin{aligned}
\sum_{v} L[s,v] \geq & P(s) \sum_{x \in S} \sum_{v} L[x,v] \\
\sum_{v} L[s,v]U(v) \geq & P(s) F_U \sum_{v} U(v) \\
\sum_{v} L[s,v]A(v) \geq & P(s) F_A \sum_{v} A(v)
\end{aligned}$$

The "fudge factors" $0 \leq F_U \leq 1$ and $0 \leq F_A \leq 1$ are input parameters to the balancing process and control the tightness of the balance. The closer these values are to 1, the more even the required distribution of usage and accesses, respectively, will have to be.

Note that this statement of the balancing constraints requires that each location take at least its "fair share" of the load. An alternative statement would be to require that each location take no more than some maximum amount of the load. This alternate statement will help prevent overfilling some locations for small values of F_U and F_A, but both statements converge into the same equation for values of F_U and F_A approaching 1.

Completing the Problem Statement

We can now say that the goal of a balance calculation is to find a modified boolean location matrix L' satisfying the constraints discussed above. To formalize our definition of a location matrix, we add one additional constraint, ensuring that a given volume is placed in one and only one location:

$$\sum_{s} L[s,v] = 1 \qquad \forall\ v \in V$$

We would prefer not to move every affected volume, though, so we add the additional instruction to minimize the value:

$$\sum_{s} \sum_{v} L'[s,v](1 - L[s,v])$$

or in other words, to minimize the total number of volumes moved. This is just one possible optimization rule; another, possibly better one would be to minimize:

$$\sum_{s} \sum_{v} L[s,v](1 - L'[s,v])(U(v) + K)$$

for some constant K corresponding to the amount of data that can be moved in the length of time required to perform the fixed costs associated with moving a volume (primarily updating the VLDB). This will minimize the length of time required to implement the balance, since the time that it takes to move a volume is proportional to its size.

Note that while the desired volume relocations can be determined by comparing L and L', a useful trick to ease that process is to instruct the linear programming optimizer to output, as its result, the matrix:

$$M[s, v] = L'[s, v](1 - L[s, v])$$

$M[s, v] = 1$ if and only if the volume v was not previously in location s but should be in location s after implementing the balance. In other words, every 1 in the result matrix M corresponds to a single volume move required to implement the solution.

Credits

This AFS server balancing algorithm was developed by Neil Crellin. I wrote the above documentation of the mathematics by analyzing the AMPL model that he developed and observing the results of implementing it.