

Distributed Storage

Development of a redundant distributed backup system

Abstract

In many computer labs, there are large numbers of computers with unused drive space. There may also be relatively large quantities of data to be backed up. The goal of this project was to develop a system for storing data distributed over many computers, with enough redundancy so that data can still be recovered if several of the machines are unavailable (due to inevitable hardware failure). It was also enhanced to provide faster distribution using distributed computing and random access to backups, allowing access to individual files.

Background

Reed-Solomon is an advanced redundancy method used by many devices such as CD-ROMS. The idea is that Raid-Solomon can be used relatively simply when you know which data is lost; normally it is used in devices such as CD-ROMS, where it is impossible to know which data is correct and which was mangled. But in "RAID-like Systems," it is known which device failed, so data recovery is simpler. In particular, this algorithm can be used for recovering data from multiple computers when some computers are dead.

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} & a_{04} \\ a_{10} & a_{11} & a_{12} & a_{13} & a_{14} \\ a_{20} & a_{21} & a_{22} & a_{23} & a_{24} \\ a_{30} & a_{31} & a_{32} & a_{33} & a_{34} \\ a_{40} & a_{41} & a_{42} & a_{43} & a_{44} \\ a_{50} & a_{51} & a_{52} & a_{53} & a_{54} \\ a_{60} & a_{61} & a_{62} & a_{63} & a_{64} \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix} = \begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \end{bmatrix}$$

Multiplying the data (d) by a matrix (a) to get a result which can be stored over multiple computers

Algorithm

The basic idea is to put the data, into an $n \times 1$ matrix, and multiply some other $n + m \times n$ matrix by it, resulting in an $m \times n$ matrix. Each row in this matrix represents the data to be sent to a certain device. As a result of this, if n devices out of the $n + m$ are available, an $n \times n$ matrix can be constructed and the resulting equation can be solved for the original data.

$$\begin{bmatrix} b_{00} & b_{01} & b_{02} & b_{03} & b_{04} \\ b_{10} & b_{11} & b_{12} & b_{13} & b_{14} \\ b_{20} & b_{21} & b_{22} & b_{23} & b_{24} \\ b_{30} & b_{31} & b_{32} & b_{33} & b_{34} \\ b_{40} & b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix} \begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix}$$

Recovering the original data using a subset of the redundant data, and appropriate matrix (b)

Procedure

This program will was done in several steps - first testing the redundancy algorithm, then moving it to multiple computers, then adding extra features such as distributed computers and random access within backups. This was done to ensure that each part was fully tested an working before another layer of complication was added; otherwise, testing and debugging would have been nearly impossible.