Project Information theory
Distributed storage

Samuel Van de Velde

1 Introduction

This document describes the project assignment for the course on information theory. In this project, the theory from the course will be applied to cloud storage. This project should be carried out by groups of 2 or 3 people and consists of writing a report (in English) and implementing a cloud storage system in Matlab. A printed copy of the report should be handed in to Samuel Van de Velde. The Matlab code should be uploaded to the dropbox on Minerva as a zip-file with the name groupXX.zip with a map called groupXX containing all the code. **The project deadline is May 22.**

2 Distributed storage for big data

Disk capacity has increased a lot in the past years: from a few megabytes (MB) of data capacity to gigabytes (GB) and now multiple terabytes (TB) on a single hard disk drive (HDD). However, the amount of data that is generated increases by 40% each year and storing this enormous amount of data requires storage systems comprising hundreds or thousands of physical hard disks, possibly spread out over a number of data-centers over the world. In systems with such a large amount of disks, the possibility of a disk breaking down is no longer negligible. For example, in Fig. 1, the amount of failing disks in a Facebook cluster of 3000 disks is shown on a daily basis, with failures ranging between 0 and 110. Protection to such failures must ensure that no data is lost. RAID (redundant array of independent disks) is a traditional storage technology that provides protection by adding redundancy to the data and storing it over multiple disks. Whenever a disk would fail, the data can be recovered by combining data from other disks. However, this storage technology is not very flexible does not scale well for very large systems. In this project we will explore storage systems where the reliability is provided by erasure codes.

The storage system we will consider is in line with that of the Belgian company Amplidata that achieves a raw data capacity of 36 petabytes (1 PB = 1000 TB). The storage is distributed over 3 data-centers worldwide (See Fig. 2). In every datacenter, two racks of 600 10TB disks (or nodes) are installed. The systems must achieve very high reliability: a single disk can fail, a full rack

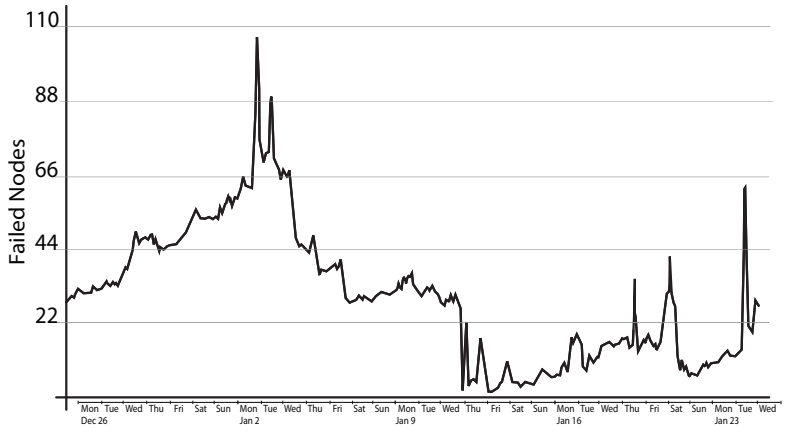

Figure 1: Number of failed nodes over a single month period in a 3000 node production cluster of Facebook.

can fail, and even a whole data-center can fail (for example as a result of power outage or a fire).

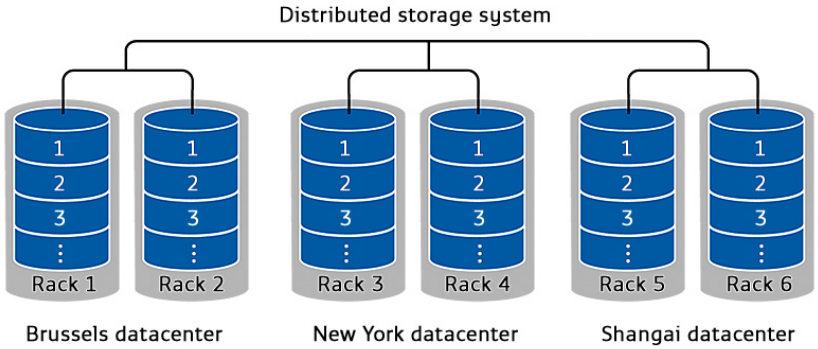

Figure 2: Schematic of the distributed storage system

3 Introduction on data protection

3.1 Erasure codes

In the theory we have seen that the Reed Solomon (RS) code is a code that requires $2t$ parity symbols to correct up to t symbol errors. Error correction is performed by solving a non-linear system of $2t$ equations (one equation for every syndrome) to obtain both the position and the value of each error. In this project we will see how we can use the RS code to recover from erasures. An

erasure is an error of which the position is known, but not the value. Because of this extra information, it becomes possible to recover $2t$ erasures with an RS code.

Consider an (n, k, t) RS code in GF(2^m). Let $c(x)$ be the original codeword polynomial and $r(x)$ the received polynomial after some error event. We can write $r(x) = c(x) + e(x)$ where $e(x)$ is the error polynomial. When we have v erasures, $e(x)$ is written as $e(x) = e_1 x^{i_1} + e_2 x^{i_2} + \ldots + e_\nu x^{i_\nu}$ with i_1, $i_2, \ldots,$ i_v the *known* positions of the erasures and $e_1, e_2, \ldots, e_\nu$ the values of the erasures which are elements of GF(2^m). By definition, the $2t$ syndromes (of the third type) can be computed as follows

$$S_{j-1} \quad = \quad r(\alpha^j) \tag{1}$$

for $j = 1...2t$. Alternatively, the syndromes are also given by using the error polynomial $e(x)$:

$$\begin{aligned} S_{j-1} &= r(\alpha^j) = c(\alpha^j) + e(\alpha^j) = e(\alpha^j) && (2) \\ &= e_1\alpha^{i_1 j} + e_2\alpha^{i_2 j} + \ldots + e_{2t}\alpha^{i_{2t} j} = \sum_{m=1}^{2t} e_m \alpha^{i_m j} && (3) \end{aligned}$$

for $j = 0..2t$, resulting in the following set of equations that are linear in $e_1, e_2, ..., e_{2t}$:

$$\begin{bmatrix} \alpha^{i_1} & \alpha^{i_2} & \cdots & \alpha^{i_\nu} \\ \alpha^{2i_1} & \alpha^{2i_2} & \cdots & \alpha^{2i_\nu} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha^{\nu \cdot i_1} & \alpha^{\nu \cdot i_2} & \cdots & \alpha^{\nu \cdot i_\nu} \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_\nu \end{bmatrix} = \begin{bmatrix} S_0 \\ S_1 \\ \vdots \\ S_{\nu-1} \end{bmatrix} \tag{4}$$

As long as $\nu \leq 2t$, this linear system has a unique solution for $e_1, e_2, ..., e_{2t}$ such that the error values can be found and the erased data can be restored.

4 Assignment

4.1 Data storage

This first part of the project are some general questions about data storage. Do some reading online to answer the following questions.

1. It is generally considered that RAID 5 systems are not suitable for larger disk sizes. Explain why. Is the same true for RAID 6?

2. Explain how nested RAID works. How many disk failures can RAID 60 handle at any given time?

3. Look up the datasheets of the WD Xe hard drive and the Seagate Kinetic. Compare both drives in terms of reliability. Explain which hard drive you

would choose for the best reliability. Compute the annualized failure rate (AFR) for both drives considering continuous operation (24/7)[1].

4. Briefly explain what object storage is. Is it possible (and how) to have an object storage system where objects have different levels of data protection (in terms of reliability against disk failures)?

4.2 Reed-Solomon protection

For data protection we will use a shortened (n=18, k=12, t=3) Reed Solomon code in GF(2^8). Every object that is stored is split into a number of information words of k bytes, i.e., k symbols in GF(2^8). Every information word is encoded in a codeword of length $n = 18$ that is split evenly over the different datacenters (see Fig. 2). In every rack, 3 symbols of the codeword are stored on 3 different disks. Because the RS code can recover $2t$ erasures, a total of 6 disks may fail at any given time.

1. Encoding.

   - Let α be a primitive element of GF(2^8). Consider the primitive polynomial given by: $p(D) = D^8 + D^4 + D^3 + D^2 + 1$. Show that $p(D)$ is a primitive polynomial with root α for the field GF(2^8).
   - Construct the generator polynomial $g(x) = g_0 + g_1x + g_2x^2 + \ldots + g_{n-k}x^{n-k}$ for the shortened (n=18, k=12, t=3) RS code. Give the coefficients g_i for $i = 0..n-k$ (in GF(2^8)).
   - Give the corresponding check polynomial $h(x)$.
   - For distributed storage systems, it is important that not only the data is distributed but also the computational workload. Fortunately, the encoding of an RS code can easily be distributed. Describe what operations must be carried out by every node to implement a *systematic* encoder of $g(x)$.
   - What is the minimal Hamming distance for this code. Use the fact that a Reed-Solomon code is an MDS code.
   - Explain why a Reed-Solomon code is -in general- more appropriate for data protection than a binary BCH code.
   - Implement the function *write_ data()* and *read_ data()*, assuming there were no errors (note again that you may not use Matlab implementations for RS coding/decoding)

2. Erasure and error decoding.

   - Whenever a disk fails, a rack fails, or a full datacenter is unavailable we use erasure decoding to recover the data. Implement the function *repairFailedNodes()* for erasure decoding.

[1] Note that, in practice the AFR is typically 5 to 10 times higher as the number stated by the manufacturer.

- Every once in a while, a random bit flip occurs on a disk called a 'bitrot' (or a bit error). To recover from these types of errors, error decoding is performed periodically in the background. This capability is sometimes called 'self-healing' or 'data scrubbing' in the context of storage systems. Implement the error decoding algorithm in the function *selfHeal()* to recover from bitrot.
- What is the minimum and maximum amount of bitrots that the (18,12) RS code can correct?
- Simulate the bit error rate after RS decoding for a varying bitrot probability p. Plot the result for values of p between $p = 10^{-2}$ and $5 \cdot 10^{-1}$ (use the function *bitRotStrikes()*). Also show the analytical bit error rate as a reference. What can you conclude?
- In one study, performed by CERN over six months and involving about 97 petabytes of data, about 128 megabytes of data became permanently corrupted by bitrot. How many corrupted bytes can we expect in our 36 petabyte storage system after one year. How many errors will remain after RS error correction? From this analysis, is bitrot a big problem?

4.3 Locally Repairable Codes

With the RS-code that we designed in the previous section, we have the problem that whenever a disk fails, the data from all other disks must be read in order to reconstruct the data from the failed disk. This slows down the storage system during reconstruction and results in high network traffic. Solving this problem is an active field of research and some solutions already exist. One solution is offered by so-called locally repairable codes (LRC). They are designed to keep the strong recovery capability of the RS-codes, while being able to reconstruct data from a few disks close to the failed disk.

A (n, k, t, r) LRC code splits n symbols into $n/(r+1)$ repair groups consisting of $r + 1$ nodes. For example, a datacenter can be a repair group. Within every repair group, a single erasure can be corrected using the r other nodes within the repair group. We call r the repair locality.

Consider an information sequence $\mathbf{b} = \left[\mathbf{b}^{(1)}, \ldots, \mathbf{b}^{(r)}\right]$ where every $\mathbf{b}^{(i)} = [b_1^{(i)}, ..., b_k^{(i)}]$ is coded into $\mathbf{c}^{(i)} = [c_1^{(i)}, .., c_n^{(i)}]$ using an (n, k, t) RS code. Using these r codewords, an additional parity word is computed as follows:

$$\mathbf{s} = [s_1, ..., s_n] = \sum_{i=1}^{r} \mathbf{c}^{(i)}$$

Notice that we are summing elements in $\mathrm{GF}(2^8)$ which is simply a bitwise XOR operation (for example: $s_1 = c_1^{(1)} \oplus c_1^{(2)} \oplus \ldots \oplus c_1^{(r)}$). Now the r codewords and the parity word is split over the repair groups. In the kth repair group we will place the symbols $c_j^{(i)}$ and s_j with $j = K + 1, ..., K + r + 1$ and $K =$

Table 1: Construction of a locally repairable code. These nodes belong to the kth repair-group of $r+1$ nodes.

	node 1	node 2	...	node r	node $r+1$
blocks of $\mathbf{c}^{(1)}$	$c_{K+1}^{(1)}$	$c_{K+2}^{(1)}$	$\cdots$	$c_{K+r}^{(1)}$	$c_{K+r+1}^{(1)}$
blocks of $\mathbf{c}^{(2)}$	$c_{K+2}^{(2)}$	$c_{K+3}^{(2)}$	$\cdots$	$c_{K+r+1}^{(2)}$	$c_{K+1}^{(2)}$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
blocks of $\mathbf{c}^{(r)}$	$c_{K+r}^{(r)}$	$c_{K+r+1}^{(r)}$	$\cdots$	$c_{K+r-2}^{(r)}$	$c_{K+r-1}^{(r)}$
blocks of $\mathbf{s}$	s_{K+r+1}	s_{K+1}	$\cdots$	s_{K+r-1}	s_{K+r}

$(k-1)(r+1)$. The placement within the repair group is as follows: on the ith row, the block $\mathbf{c}^{(i)}$ is circularly shifted to the left by $i-1$ positions (for $i = 1..r$). On row $r+1$, $\mathbf{s}$ is circularly shifted to the left by r. The placement of the symbols in the kth repair group is shown below:

1. Give the table representing the data stored in the second repair group of a locally repairable code with $n = 18$ and $r = 5$. This will correspond to the data saved on the New York datacenter.

2. Give a formula for the repair of any $c_j^{(i)}$ in the kth repair group. In the table from the previous question, highlight the elements required to recover $c_{K+1}^{(2)}$.

3. Construct your own code, based on the principle of the LRC described above, that is able to locally recover 2 errors within a repair group. This time, the repair locality r is the number of nodes needed within a repair group to recover 1 or 2 erasures. Give explicit formulas for all the additional parity symbols you need. Give a table (similar to Table 1) of where the symbols are stored within a repair group. Explain briefly how data is recovered in case of one or two failed nodes.

4. Consider a raw storage capacity of 36PB, plot the actual storage capacity on a graph with on the x-axis the repair locality r and using $k/n = 2/3$. Plot the actual storage capacity in case of: i) an (18,12) RS code ii) the LRC code and iii) your code from the previous question. What is the effect of the repair locality r?

5 MATLAB implementation

It is *not* allowed to use the encoding and decoding functions provided by matlab. As a basis of the matlab implementation, some files are provided on Minerva. In the main file *TheCloud.m* you will find a class with functions that are required for a distributed storage system. You can implement these functions either directly in this main file or by using external files with the name of the

corresponding function. In the latter case, the function must be declared as follows inside the *methods*-block of the main file:

```
[output1, output2] = functionName(input1, input2);
```

It is important that he functions that must be completed use the correct input and output. This is always described in detail in the functions. This is necessary such that the files you submit can be tested for proper functionality. Notice that all files will also be tested for plagiarism. In table 2, a few useful matlab commands are shown. Make sure to provide sufficient comment in your code.

FUNCTION	DESCRIPTION
reshape()	Reshapes a matrix.
bi2de(), de2bi()	Allows to switch between binary and decimal notation. In this project you will mostly have to use *'left-msb'* as a parameter.
gf()	Useful for working in a Galois Field.
clear classes	When you have adjusted a class, it is important to clear the old class from memory. this is important in Matlab 2007 if you work with static methods or properties.
mod()	Modulo operator
conv(), deconv()	Useful for multiplication or division of polynomials.

Table 2: Some useful matlab commands.