Optimizing Replication of Cloud Storage Providers with Erasure Coding

College of Science and Technology, Temple University

Jake Roemer, Mentor: Dr. Chiu C. Tan

Problem:

As data grows so does the need for cloud storage services by companies. How do we make sure these companies can trust their data will be safe, available, and inexpensive?

Solution:

Split data between different cloud providers to prevent vendor lock-in and guarantee fault tolerance of their data. We split data between *n* different cloud providers, with a fault tolerance of *n-m,* where *m* is the number of providers holding only data information; thi method is also known as erasure coding.

Design:

Test multiple coding methods to determine which would be best for a companies needs

Coding Methods:

Classic Reed-Solomon, Strict Replication, Cauchy Reed-Solomon, Simple Regenerating Codes, Rotated Reed-Solomon

Cauchy Reed-Solomon:

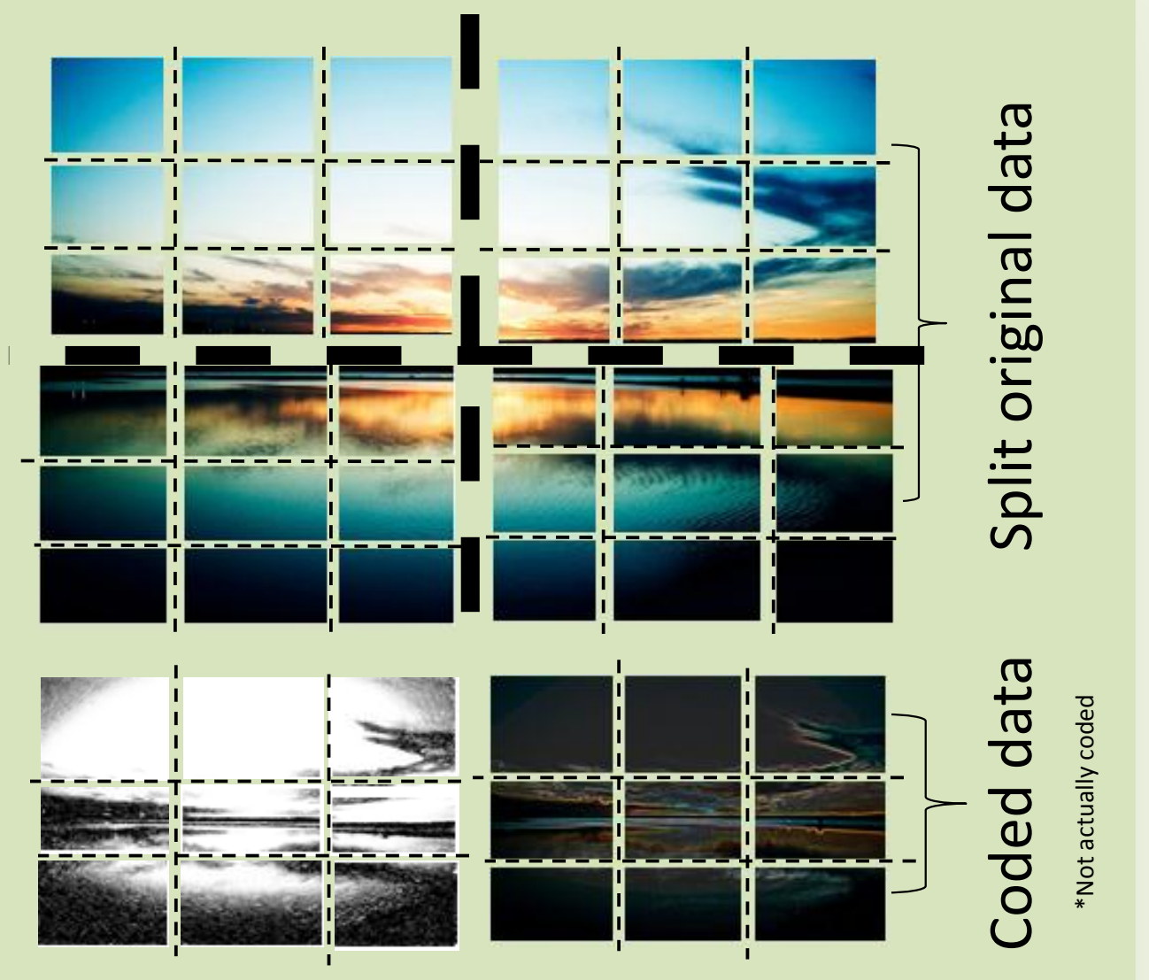
expands classic RS matrix into a *w*(n+m) x w*n* matrix which in turn expands data shares and coding shares by *w.*

Strengths:

- Only need to use XOR operations which are much less expensive than multiplication

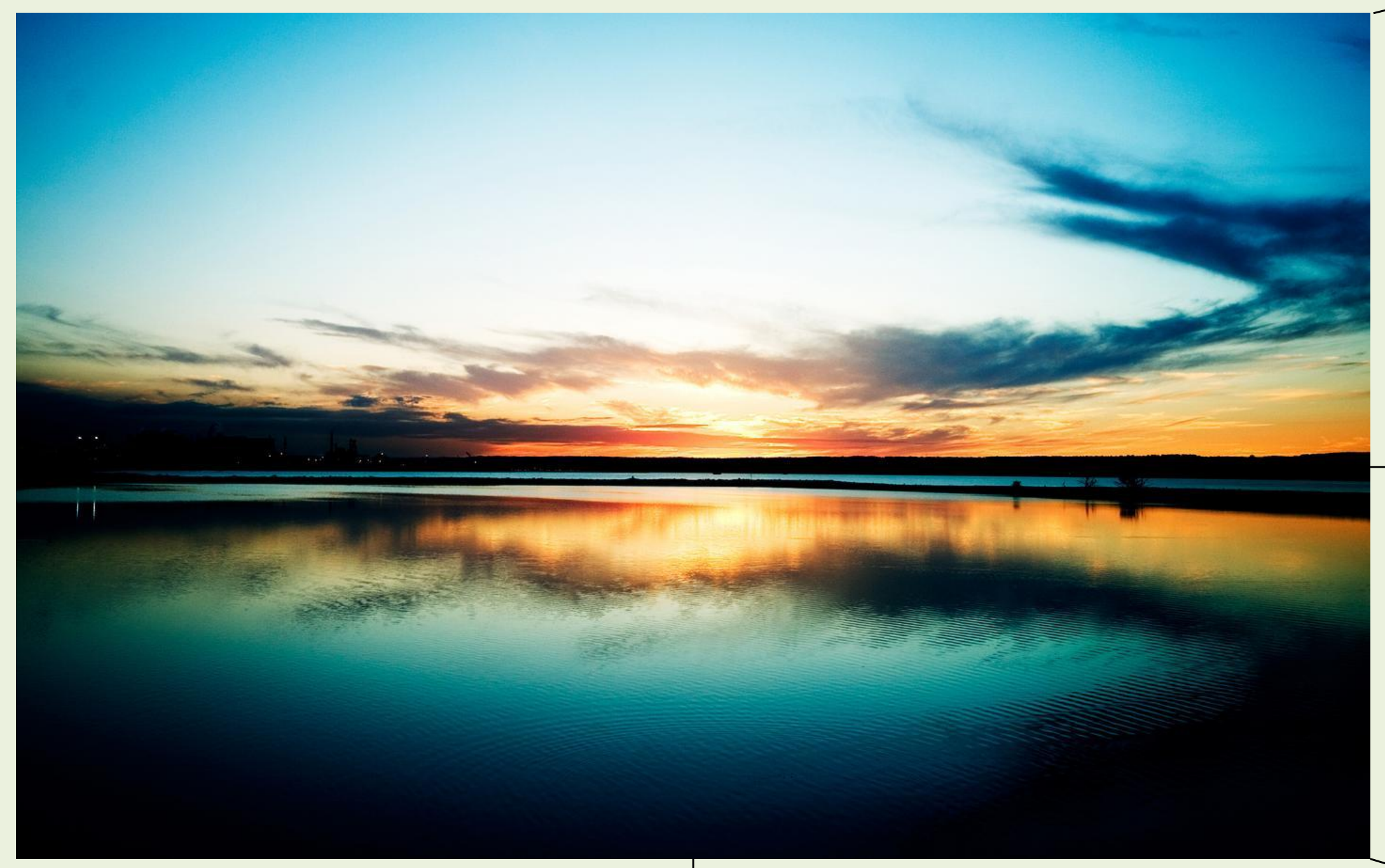
Weaknesses:

- Not all Cauchy matrices improve performance Can only withstand 1 failure
- Cannot correct errors only erasures



Replicating across 6 storage providers with a bit word size of 3

Original Data



Replicating across 6 storage providers

Replicating across 6 storage providers

Replicating across 6 storage providers

Replicating across 6 storage providers

Simple Regenerating Codes:

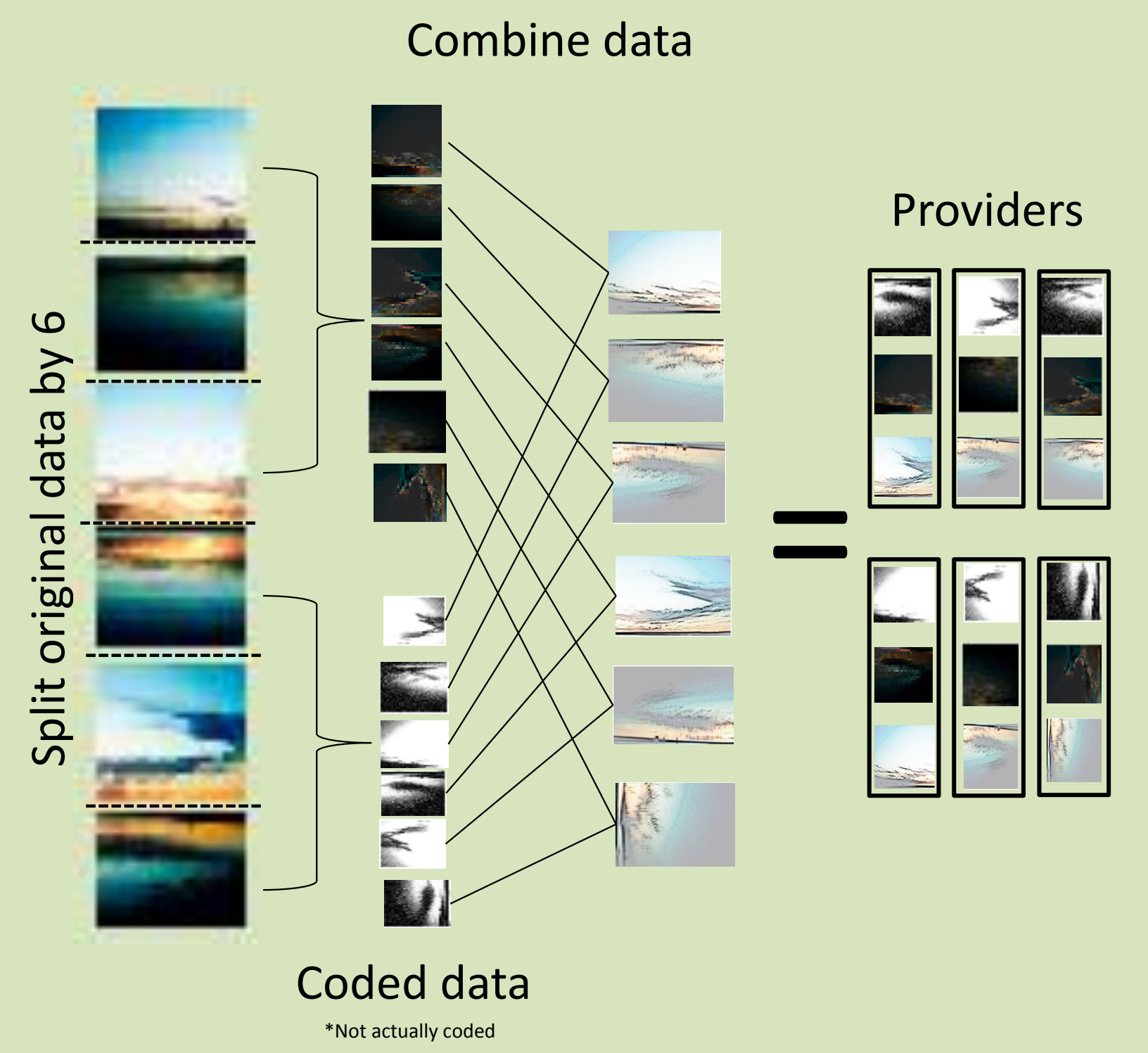
splits data into 6 shares and then splits each pair of shares into 6 more shares which undergo exclusive-or to create the coding shares, then 1 of each share is placed into one of 6 providers.

Strengths:

- Fast and efficient recovery from failed nodes
- Saves a lot of computation normally used to reconstruct original data

Weaknesses:

- Takes more time to encode data
- Increases encoding/decoding costs



Strict Replication:

makes exact copies of original data.

Strengths:

- Best Protection from data loss
- Can withstand n-1 failures
- Easy recovery from any storage node

Weaknesses:

- Very expensive to store and move data

Rotated Reed-Solomon:

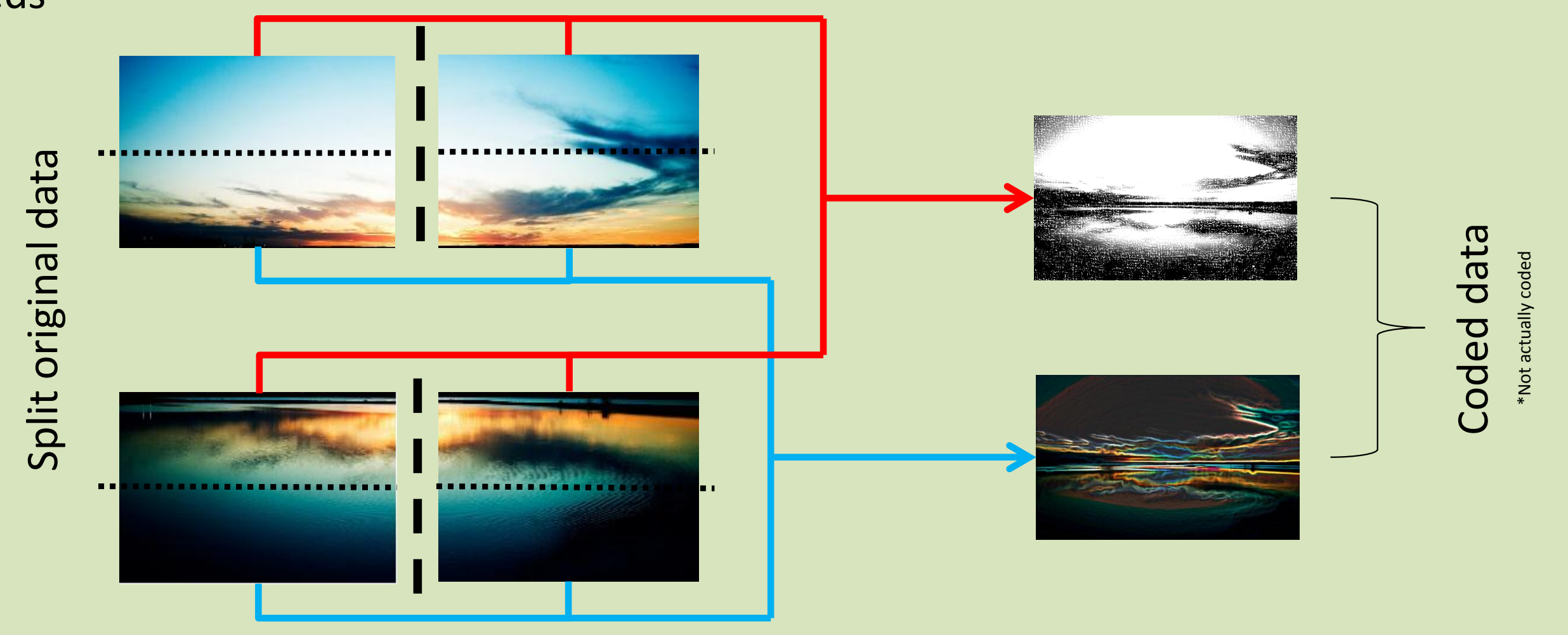
expands classic RS stripe by equation 1 to obtain a better diversity of connection between different shares.

Strengths:

- Improve performance of degraded reads
- Data disks are dependent of each other

Weaknesses:

- Is only an improvement for one disk failure at a time
- To reconstruct coding disks, rotated RS still needs the same amount of symbols as classic RS needs



Classic Reed-Solomon:

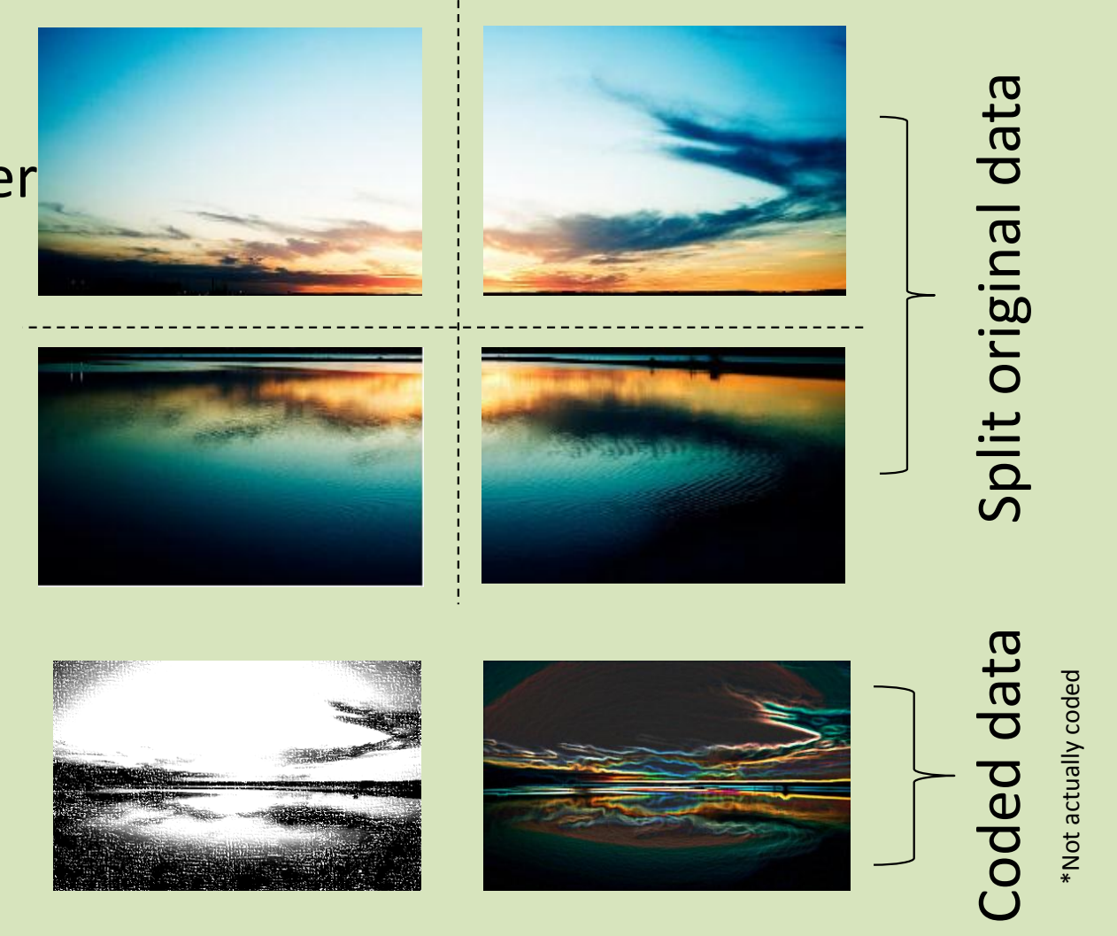
splits data into m data shares and k coding shares which can recover from any k failures.

Strengths:

- Easy to understand, Can use any n and m
- Is MDS or Maximum Distance Separable which means it can withstand *n-m* failures
- More than one way to do multiplication

Weaknesses:

- Multiplication is very expensive
- *w* must align on word boundaries



Analysis:

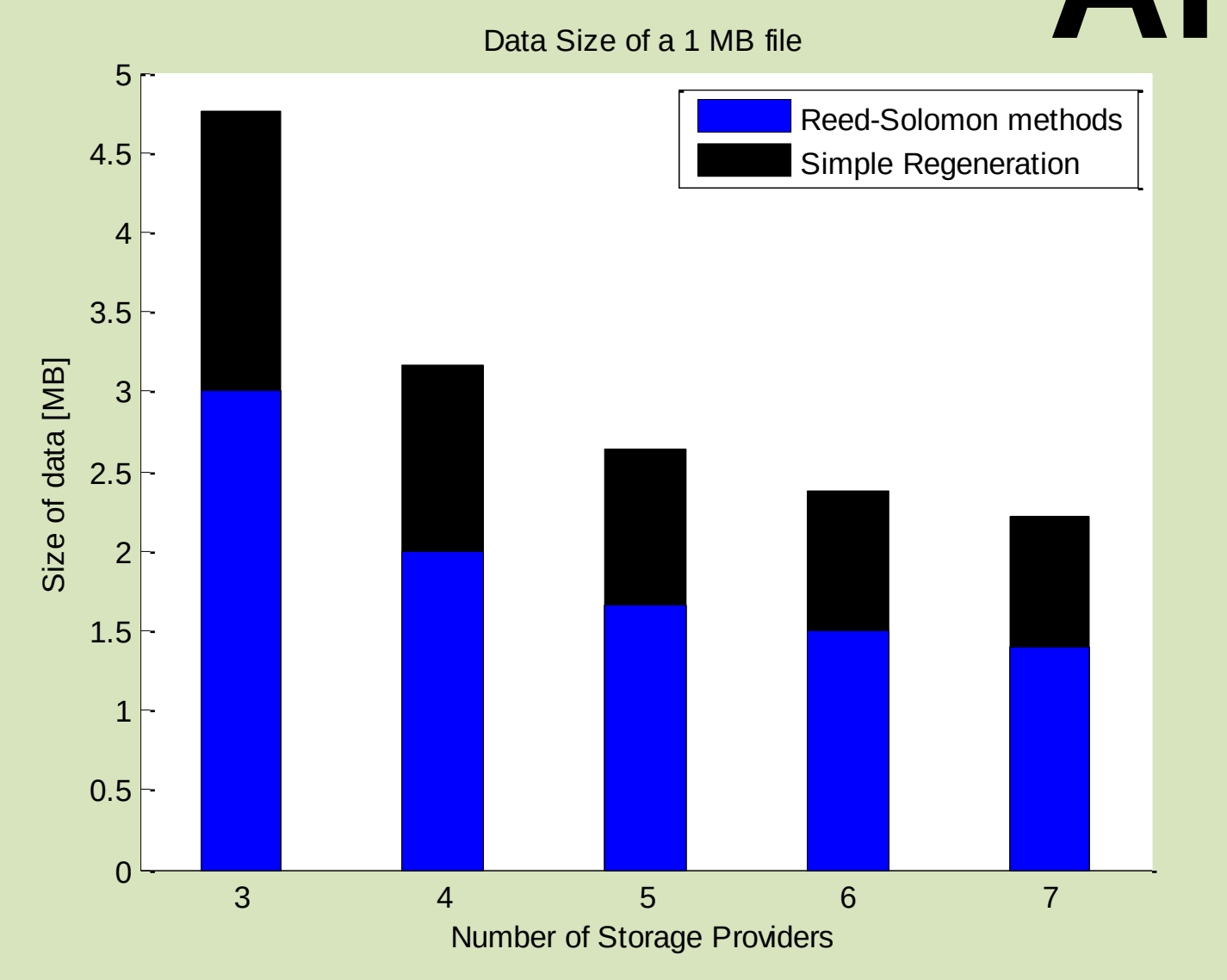


Figure 1: Shows how much data will be stored based on the total number of providers used. In each case the fault tolerance is 2 and Classic, Cauchy and Rotated Reed-Solomon methods have the same impact on amount of data stored. Strict Replication is a one-for-one ratio, so with *n* storage providers data to be stored increases by *n*.

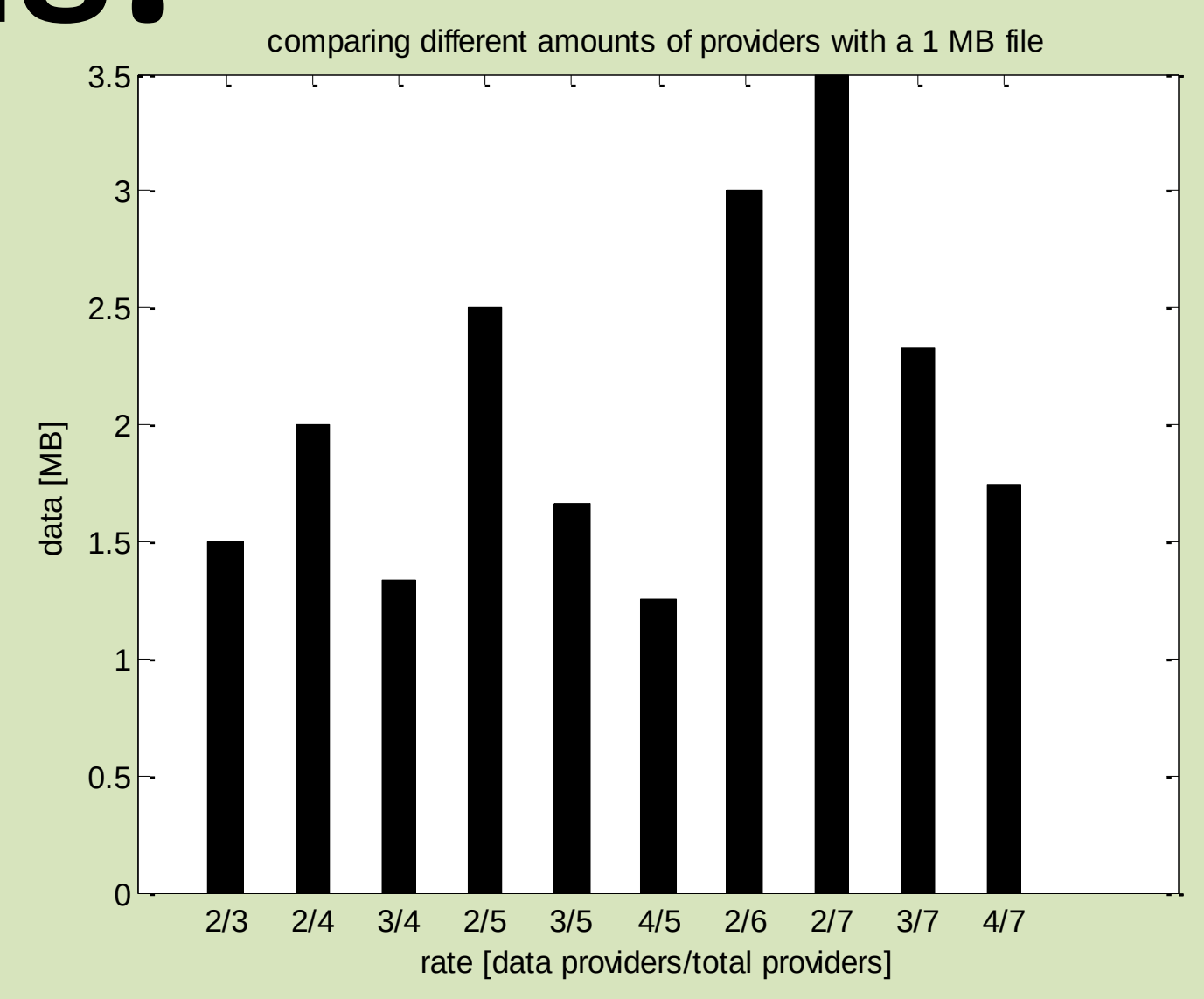


Figure 3: Shows different ratios of data only providers/total providers, also known as rate. The rate increases the amount of data being stored by 1/rate so more coding providers is less beneficial than more data providers. As long as the data providers stays close to the number of total providers having more total providers can result in a smaller cost to store data.

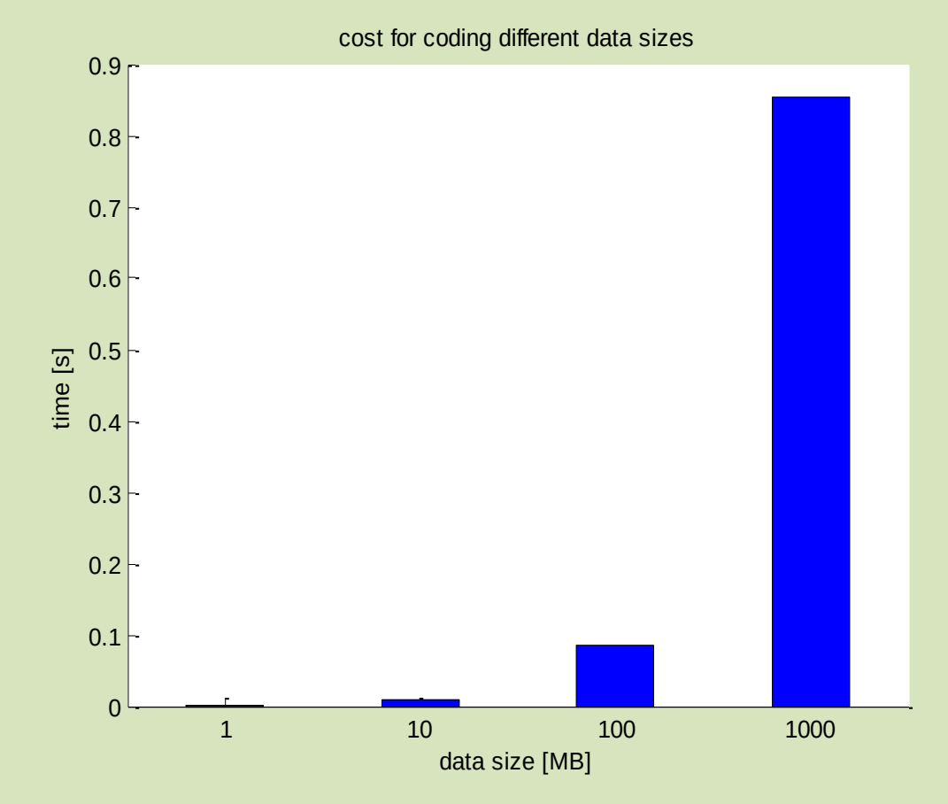


Figure 2: Shows on average how long it takes to code different sizes of data with each coding method. Each method takes a different amount of time to code each size of data as seen in figure 4 but coding different sizes of data scales the same for each method.

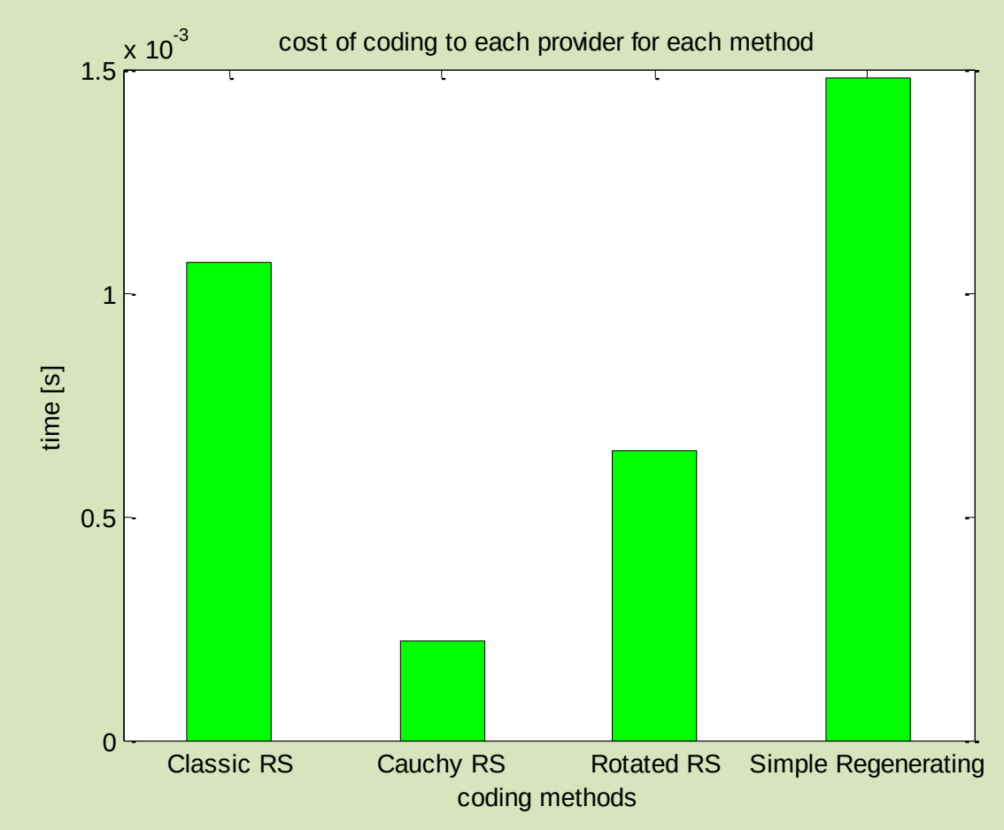


Figure 4: Shows how long it will take to code to any one provider using each method. Strict Replication does not need coding providers so the cost is zero.

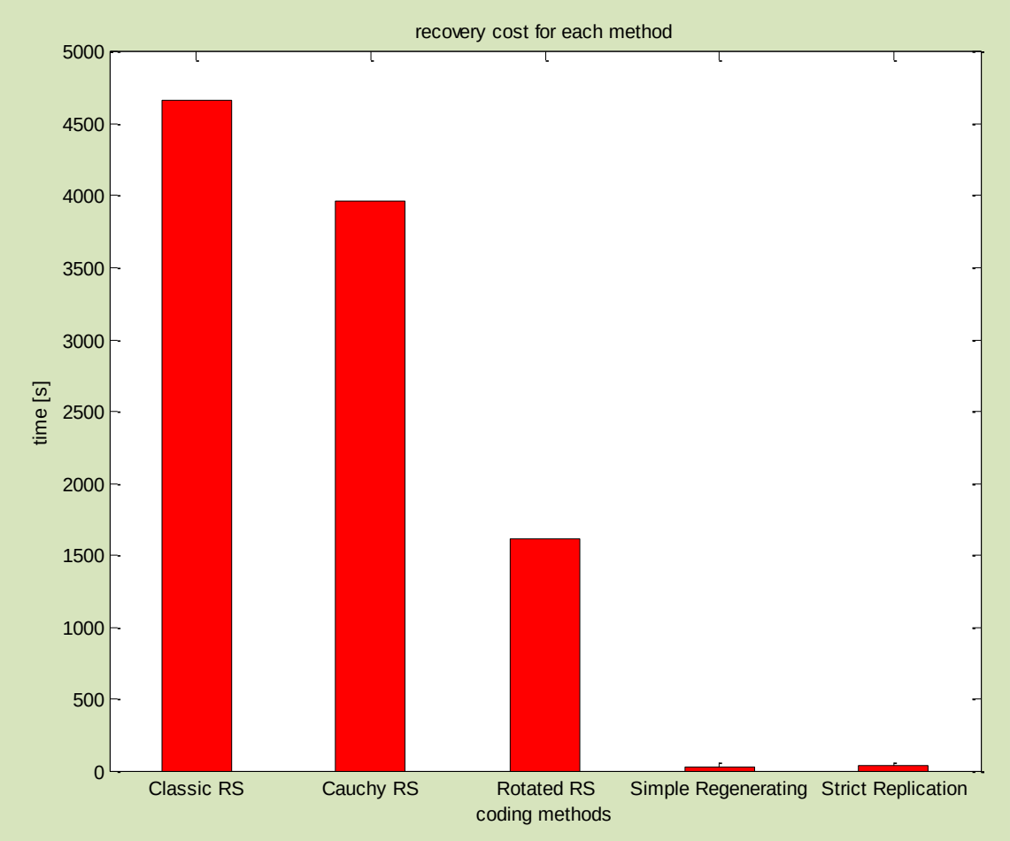


Figure 5: Shows the cost to recover from a failed provider. We used an average of randomly failed providers from 0 failed up to 2 failed.

Conclusion:

- **Strict Replication** should be used in companies which need the greatest fault tolerance and can overlook high cost.
- **Classic Reed-Solomon** is the easiest method to implement and is a vast improvement over strict replication in terms of cost. Companies who need a large fault-tolerance guarantee with safety from erasures and errors should pick Classic Reed-Solomon.
- **Simple Regenerating Codes** will work for any company in need of cheap recovery. Metrics can change depending on how you implement regenerating codes but combining coding data and original data reduces the cost of recovery by about 83% of methods like Classic Reed-Solomon.
- **Rotated Reed-Solomon** will perform as other Reed-Solomon methods in terms of encoding/decoding and recovery. Replication between storage providers can mean slow providers are seen as failed providers. This can lead to out-of-date data or frequent recovery costs. Rotated Reed-Solomon lowers the cost of recovering data to prevent out-of-date data getting to the user.
- **Cauchy Reed-Solomon** is generally a better and faster method over Classic Reed-Solomon in terms of encoding/decoding and recovery but this method is much harder to work with and understand. There are also extra measures taken to make sure the Cauchy Reed-Solomon you are using will be beneficial in the end; and this method can only withstand one failure.