The rsync algorithm

https://rsync.samba.org/tech_report/tech_report.html

An easy problem

- I have two files A and B. I want to make B equals to A

- What is the cost?
  - CPU
  - Data moved (reads, writes)

STOP The problem of rsync

- A is stored in computer **alpha** and B in computer **beta**
- The network link can be slow (at least it is much slower than CPU)

- **How can I save bandwidth?**

A naïve approach

- **Beta** compute a hash of the file B and send it to **alpha**
- **Alpha** compute the hash of A and send back to **beta** either the hash (if the two hash are the same) or the content of A if they differ
- **Beta** check if the message is the hash or has to update B

- What is the cost?
- What is the hash function?

Cryptographic hash

1. Deterministic
2. Quick to compute
3. Infeasible to generate a message from the hash
4. A small change in the message should drastically change the hash
5. It is infeasible to find collisions

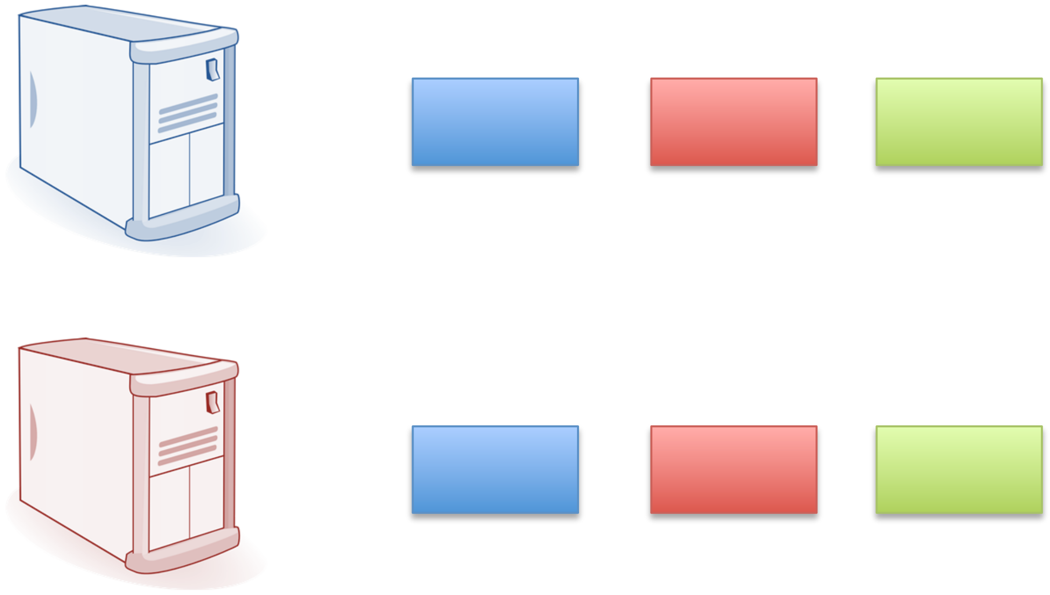
Cryptographic hash

1. Deterministic
2. Quick to compute
3. Infeasible to generate a message from the hash
4. A small change in the message should drastically change the hash
5. It is infeasible to find collisions

Can I do better?

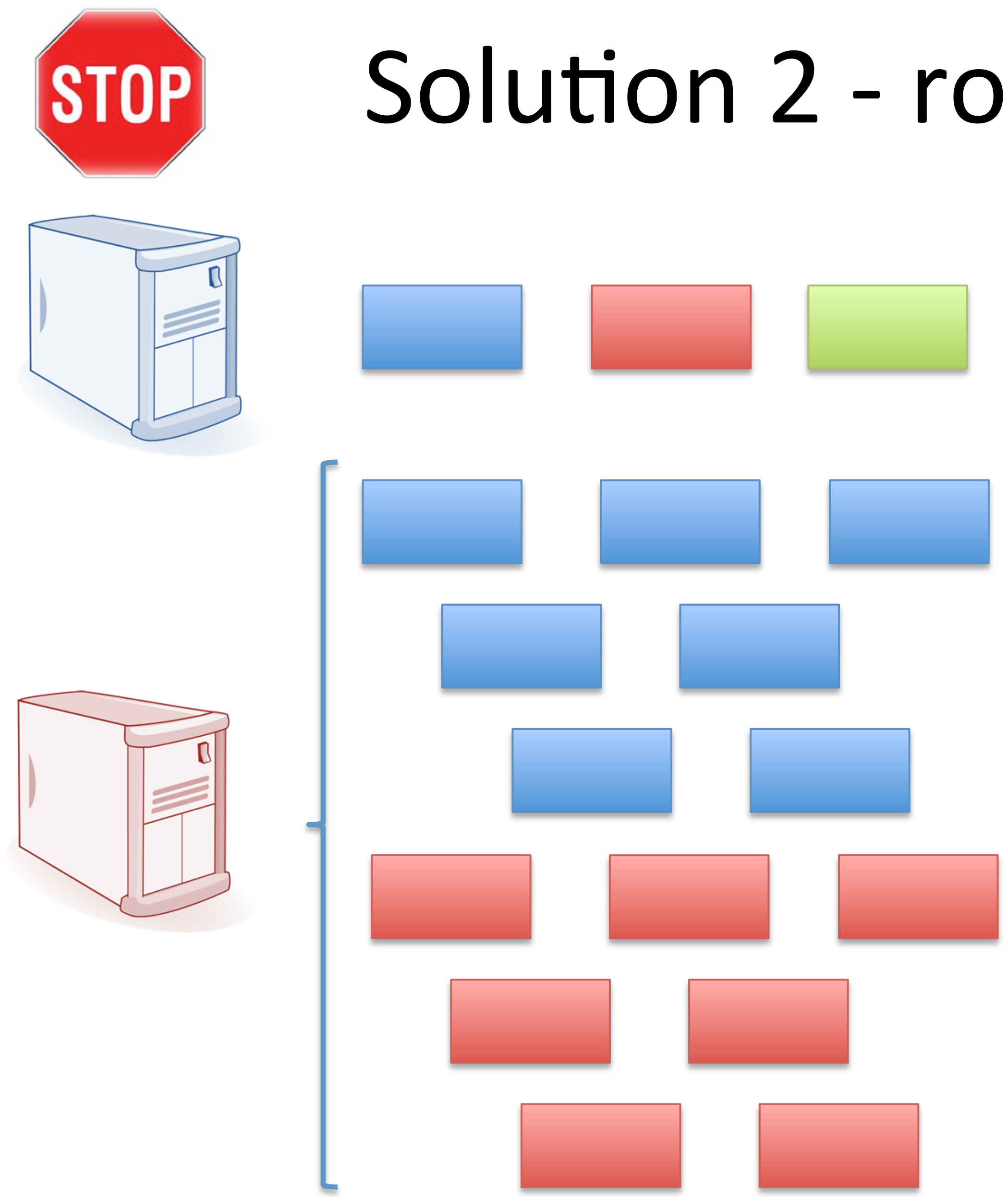
- Can I save bandwidth when A and B are similar?

Solution 1 - bucketing



- Weakness?
- Can I do better?

Solution 2 - rolling



…and the green ones as well

Can I do better?

- Intense use of cpu in **alpha**

Solution 3 – rolling hashing

- A two hashing strategy

$$Document = X_{1,}X_2 \ldots X_n$$

$$a(k,l) = \left(\sum_{i=k}^{l} X_n\right) mod\ M$$

$$b(k,l) = \left(\sum_{i=k}^{l} (l-i+1)X_n\right) mod\ M$$

$$s(k,l) = \ a(k,l) + \ 2^{16}\, b(k,l)$$

Solution 3 – rolling hashing

- A convenient way to derive next hash

$$a(k+1, l+1) = (\, a(k,l) + \ X_{l+1} - \ X_k)\ mod\ M$$

$$\begin{aligned} b(k+1, l+1) \\ &= \ (b(k,l)X_k - (l - k + 1) \\ &+ a(k+1, l+1))\ mod\ M \end{aligned}$$

- Is it M=2^{16} a good idea?
- Collisions?

Questions?

1. What is the difference with the Rabin fingerprint?
2. What is the difference with the KarpRabin searching algorithm?

Solution 4 - rsync

- Use two hash functions
- The rolling hashing for each possible offset
- A stronger 128bit hash in case a collision is detected
  - Rsync uses MD4

Solution 4 - rsync

- Use two hash functions
- The rolling hashing for each possible offset
- A stronger 128bit hash in case a collision is detected
  - Rsync uses MD4

- How to generate collisions in MD4
  - https://eprint.iacr.org/2005/151.pdf

Checksum searching

- **Beta** sent several checksums
- For each test **alpha** performs a search on these checksums

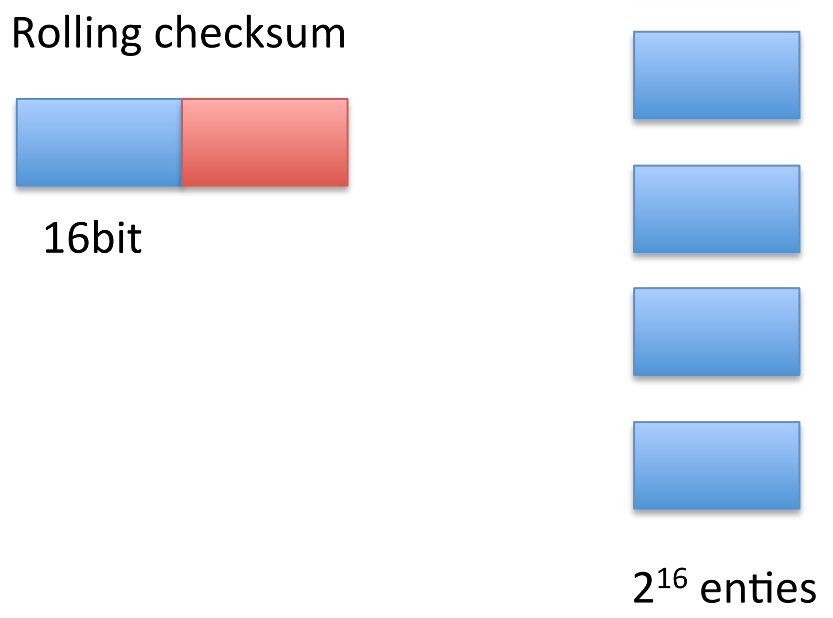
- Is linear scanning an option?

Checksum searching

- **Beta** sent several checksums
- For each test **alpha** performs a search on these checksums

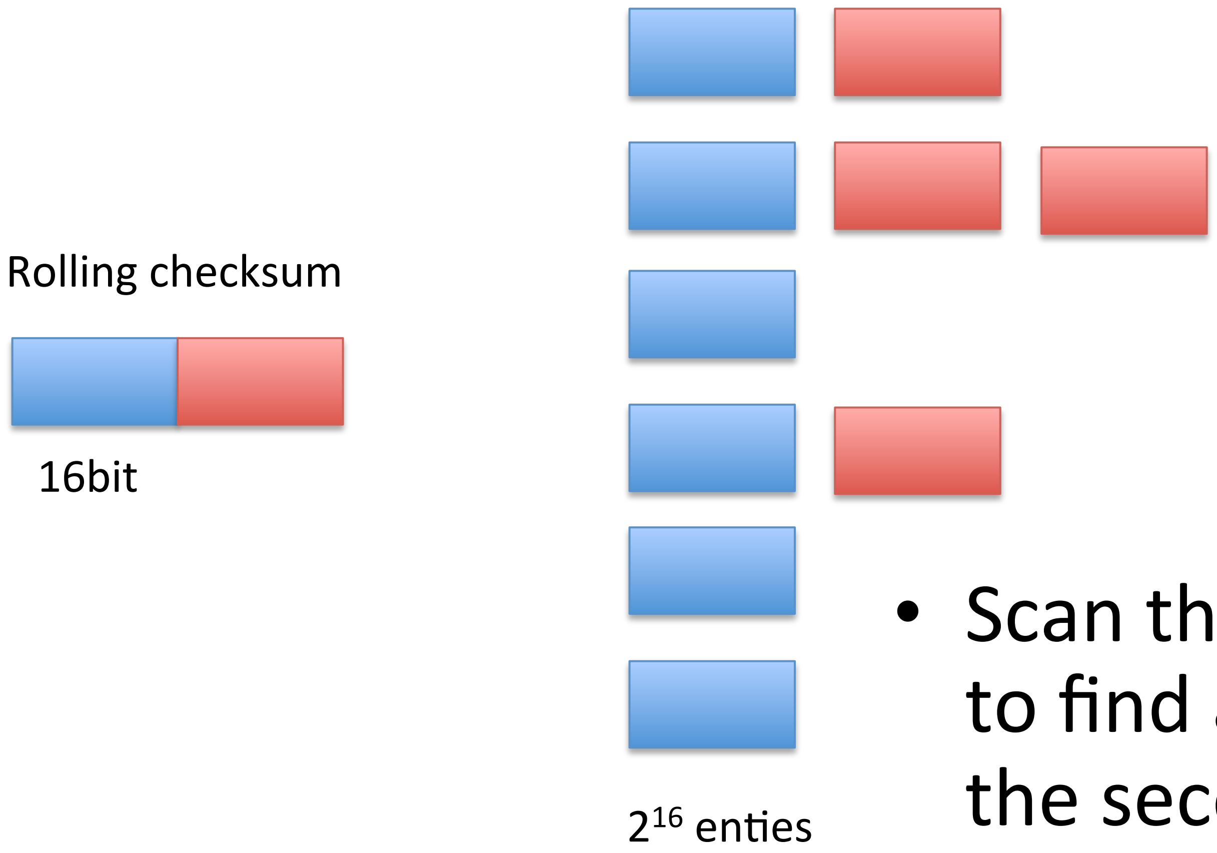
- Is linear scanning an option?
- Binary search
- Perfect hashing
- What is the preprocessing and querying cost in terms of CPU and memory?

The rsync three way test



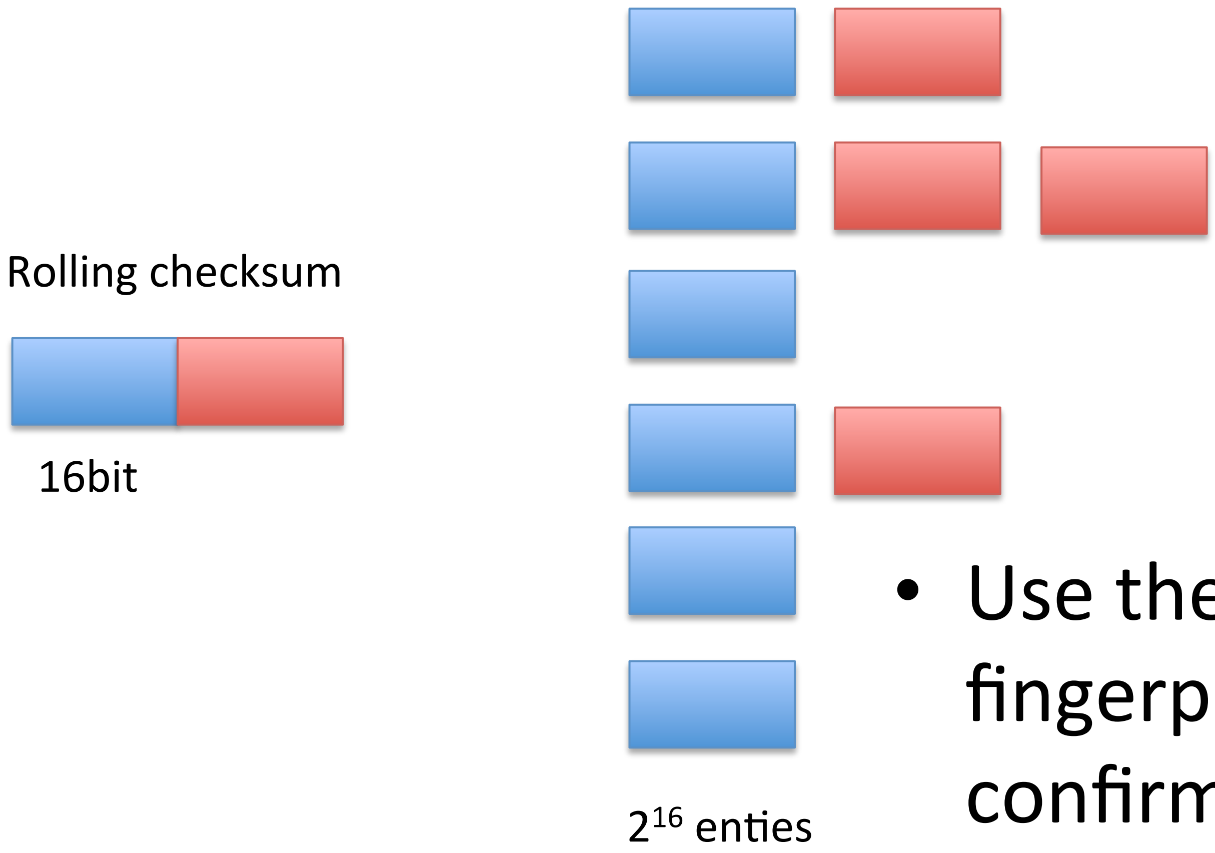
- Search for a match in the table
  - If nul the block is not found

The rsync three way test



- Scan the sorted list to find a match with the second half of the checksum

The rsync three way test



- Use the strong fingerprint to confirm the match

The rsync three way test

- What happens if two blocks in B have the same fingerprint?
- How the list of blocks can be organized?
- Is it possible to copy a corrupted file?

Things you may want to try and discuss next week

- Test the karpRabin algorithm
- Test binary search or perfect hashing
- Test the impact of the length of the block

- Small vs huge files