Where'd My Photos Go? Challenges in Preserving Digital Data for the Long Term

Professor Ethan L. Miller
Storage Systems Research Center
University of California, Santa Cruz

ssrc Storage Systems Research Center
CRSS
The University of California · Let There Be Light · 1868
Baskin Engineering UC Santa Cruz

What does “preserving data” mean?

- Preserving the actual information
  - Ensuring that the information can be read later
  - Periodic refreshes: information, media, etc.

- Preserving the *meaning* of the information
  - Ensuring that future generations can *understand* the information
  - Not sufficient to simply preserve bits!

- Some functionality is a bit of both
  - Integrity of information

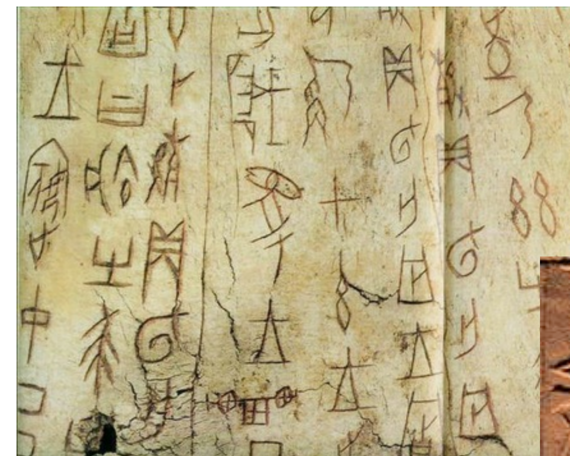
Why is digital data preservation an important problem?

- Our civilization's legacy is passed on to future generations by physical means
  - Information isn't encoded in our genes

- Historically, information was analog
  - Oral
  - Written
- For modern society, information is ***digital***
  - We need to shepherd digital data to preserve information
  - Digital data poses unique challenges

Preserving data has long been a challenge

- Ancient peoples wanted to pass down information
  - Originally, used verbal transmission: integrity issues
  - Physical transmission was more reliable
- Data was analog, not digital
  - Many lessons for preserving digital data...
- Several issues
  - Media reliability & readability
  - Data integrity
  - Preserving the meaning of the information



Media reliability

- Some media are more reliable than others
  - Paper is unreliable: must be constantly recopied
  - Parchment is more reliable, but still vulnerable
  - Stone can be very reliable
    - If nobody deliberately erases it!
- Media vulnerability mitigated by copying
  - Constantly recopy information to ensure survival
  - Problem: integrity

Data integrity

- Lots of copies ➔ potential errors
  - Make independent copies?
  - Complicate the material?
  - Rules for copying?
- All of these techniques were designed to ensure integrity of information
  - Problem: integrity may require understanding
  - How can you know that it's wrong if you don't know what it means?

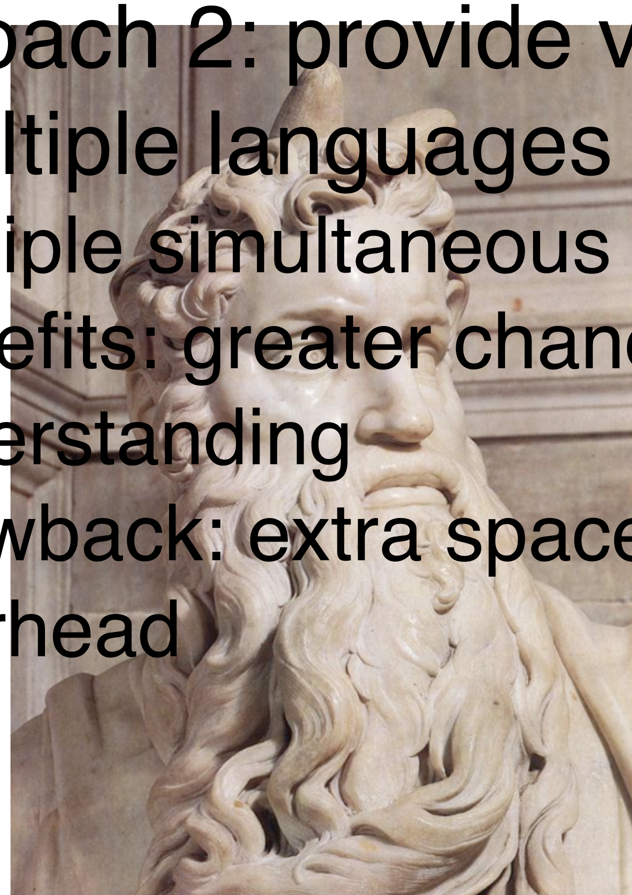
Preserving meaning

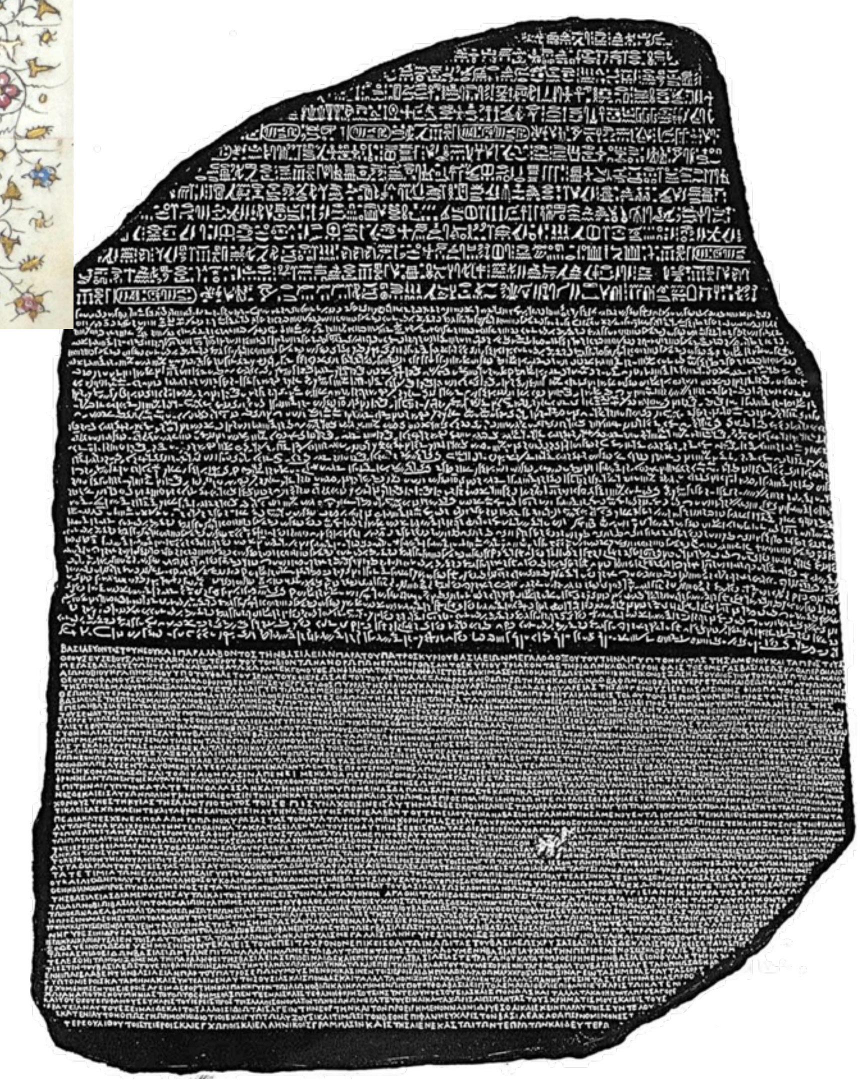
- How can meaning be preserved?
  - We often assume that languages remain static
  - We often assume that symbols remain static
- Over long periods of time, *everything* changes
  - How can we allow future users to read our data?
  - Several possible solutions...

Preserving meaning over time

- Approach 1: translate during copying
  - Widely used for many texts
  - Benefit: always have a currently-readable version
  - Drawback: errors in translation

- Approach 2: provide versions in multiple languages
  - Multiple simultaneous versions
  - Benefits: greater chance of understanding
  - Drawback: extra space overhead





Preserving digital data

- Digital data has many of the same issues as analog data
  - Need to preserve the actual bits
    - And be able to read them!
  - Need to guarantee integrity of the information
  - Need to preserve the ability to interpret the bits
- May also need (want?) other features
  - Secrecy
  - Authenticity & provenance: link the information to a particular party
  - Scalability
  - Indexing and searching

Preserving the bits: use long-lived media

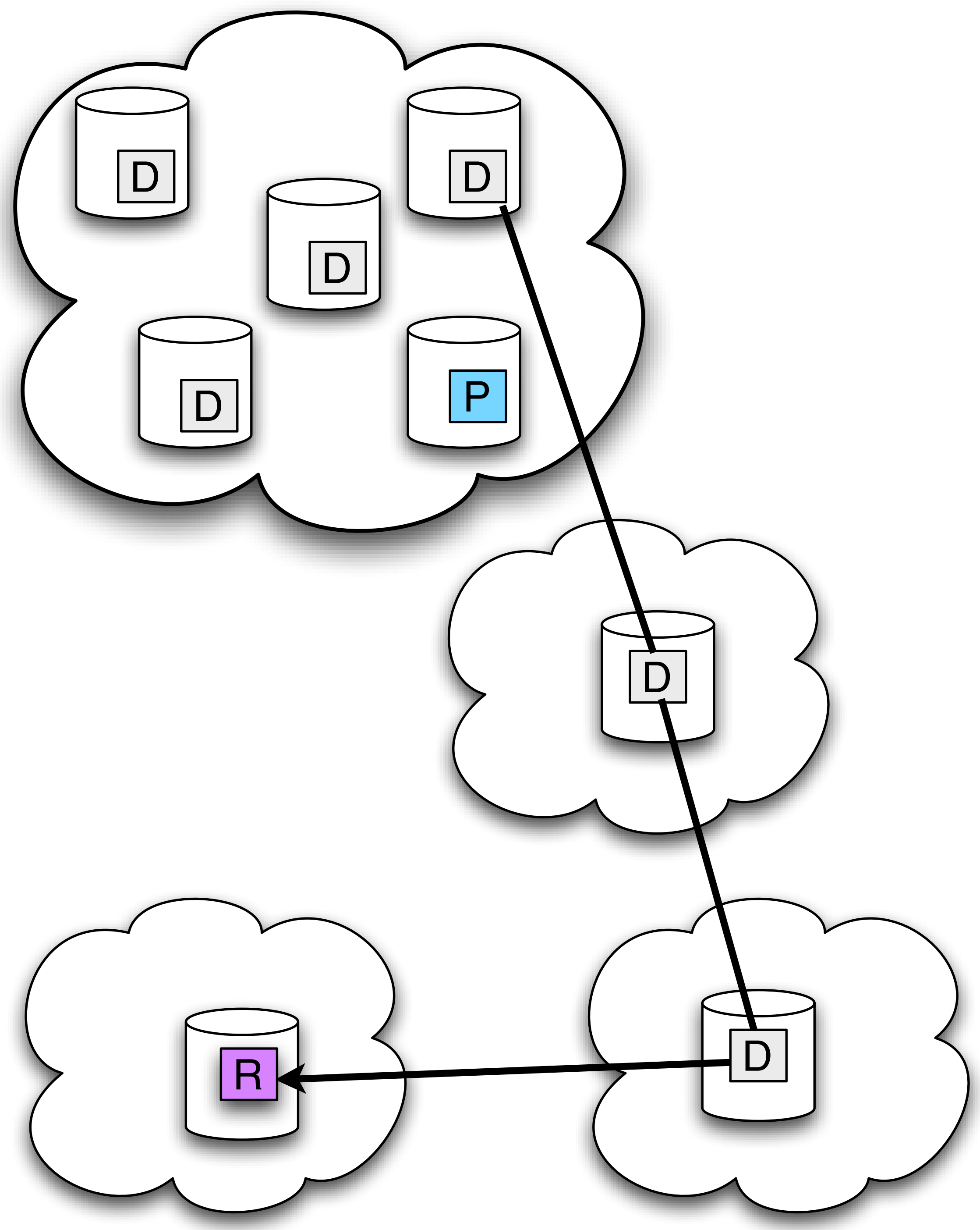
- Long-lived media work for analog data: why not use this approach for digital data?
- Inscribe bits on a stable medium
  - Use ion-beam etching to write on a stainless-steel medium
  - Information is readable with a powerful microscope
  - Information is stable for centuries to millennia
- Use magnetic tape
  - Not as stable as stainless steel
    - May last for 50+ years, but not for centuries
  - Requires more specialized hardware for reading
    - Not trivial to build a tape reader for a modern tape!
- Maybe use flash memory?
  - More on this a bit later

Preserving the bits: copying

- Making digital media last a long time is difficult!
- Alternative: use more active archives
  - Frequently (relatively) copy data to new media
- Benefits
  - Data is always on devices that can be read
  - Data can be checked for integrity during copy
  - Systems can take advantage of advances in storage technology
- Drawbacks
  - Lots of data to copy
  - May require more resources: need to refresh technology
  - Requires ***active*** participation

Preserving the bits: reliability

- Accidents will happen: bits will be lost
  - Digital data often lacks redundancy
  - Moral: keep extra copies
- Issues
  - Extra copies can be expensive
  - Extra copies need to survive “site disasters”
- Our approach: use disaster recovery codes!
  - Can be difficult to preserve metadata over time...



Preserving the bits: device evolution

- Devices change over time
  - Higher capacity
  - More reliable
  - Faster?
- Need to integrate new devices into the system
  - Can't just migrate en masse
  - Need to cope with multiple generations of devices
- Use intelligent devices
  - Networks evolve slowly
  - Internal details can be kept hidden: better compatibility

Data integrity

- Archives need to ensure that data that's read is the data that was written
  - Guard against accidental modification
  - Guard against intentional modification (rewriting of history)
- Useful to have separate independent "spheres of control" to avoid single point of failure
  - A single corrupt node can corrupt everything it manages
  - A single point can be attacked by an intruder who wants to change the world (retroactively)

Scalability

- Archives need to grow organically
  - Impossible to build initial archive at scale
  - Devices will age and die ➔ new devices will replace them
- Archives must function at small scale
  - "Minimum size" must be a few dozen devices
- Archive must scale to hundreds of thousands (millions?) of devices
  - A million disks is only an exabyte of data
  - Demand for capacity is growing very rapidly!
- Reconciling these two needs is a difficult challenge

Indexing and searching

- Analog data: small amounts ⇨ not much searching
  - But even small amounts require searches!
  - Many existing techniques: card catalogs, librarians, etc.
- Digital data is much larger!
- Indexing and searching must be
  - Efficient
  - Scalable: single large index won't work
- Self-contained media & index seems like a good approach
  - More reliable: no single point of failure
  - How can millions of self-indexed media be efficiently searched?

Long-term data secrecy

- Encryption (symmetric and public key) may be broken over time
  - Increased computing power
  - Better algorithms
  - New techniques
- Long-term secrecy needs to deal with this
  - Periodically re-encrypt
    - Difficult to do for petabytes of data
  - Use authentication instead of encryption
    - Need to guard against insider attacks
    - POTSHARDS...
- Long-term security is a big problem!

Goal: build a secure, scalable, searchable archival storage system

- Leverage earlier work done by our group: leading architectures for archival storage
- Pergamum: scalable disk-based archival storage
  - Low-power architecture built around network-CPU-flash-memory-disk nodes
  - Strong guarantees of integrity via checksumming and scrubbing
  - Error handling at both local (disk) and archive level
- POTSHARDS: secret-split archival storage to avoid single points of compromise

Who are we afraid of?

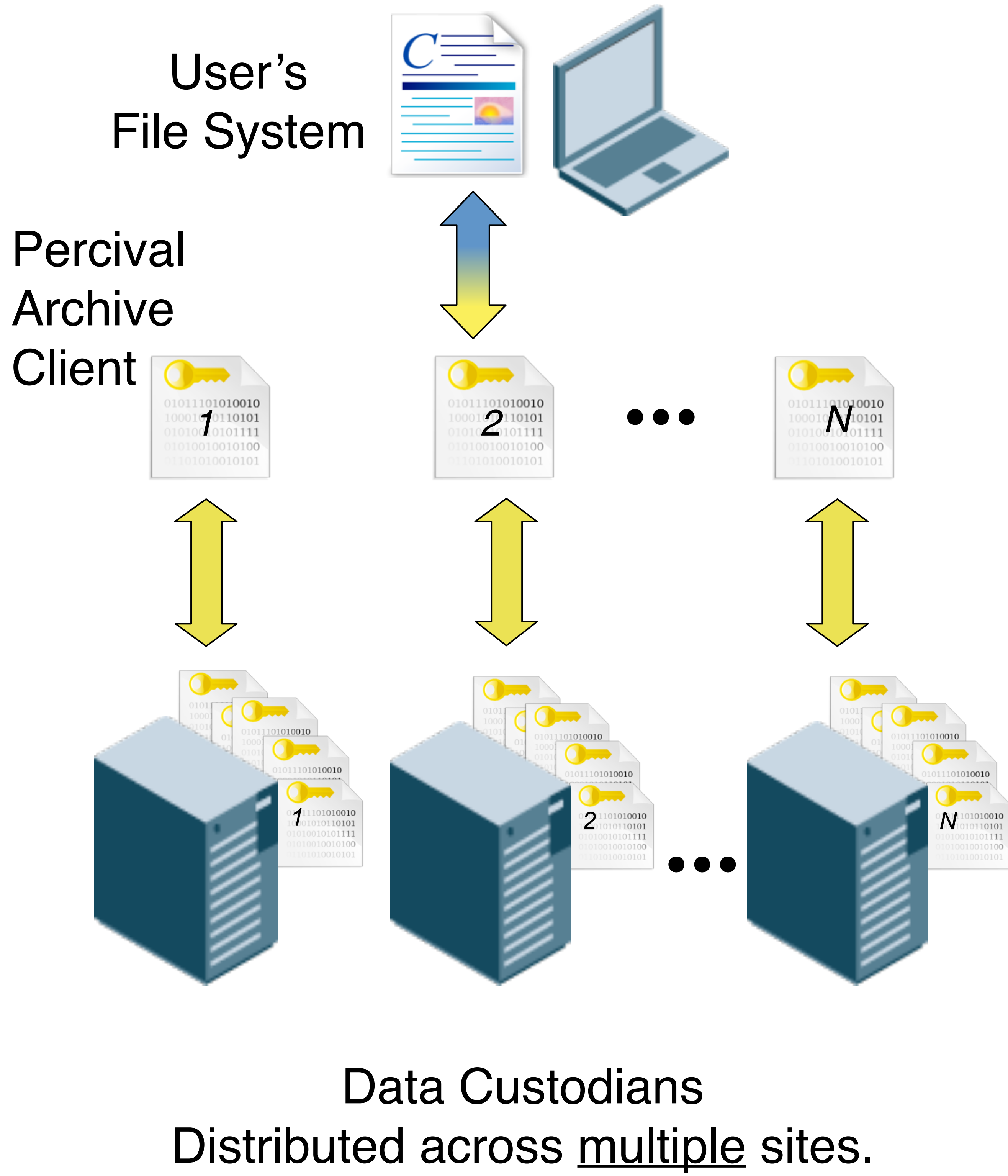
We need to reconcile our needs for privacy and utility for long-term data storage!

Threat model

- Attacker has
  - Unlimited computing power / storage
  - Unlimited time
  - Full access to any compromised repository
  - Ability to save past queries to compromised repositories
- Assume *M*-1 repositories have been compromised
- Compromise of authentication mechanism is outside of scope
  - But it's straightforward to change authentication mechanism without touching all of the data!

Challenge 1: store the data

- Use secret sharing to generate shares
- Distribute shares to each of *N* archives
  - Need at least *M* shares to rebuild
  - *N* and *M* are configurable
- Require authorization to return data to requester
- POTSHARDS and other systems do this
  - Still need work to reduce overhead of splitting



How does this help?

- No "information" at any one site
  - Must compromise *M* sites to gain any useful information
  - Difficult to do this undetectably
- Immune to key loss
  - Archives can pool their shares to allow rebuilding of data
- Immune to key / encryption algorithm compromise
  - Many forms of secret splitting are information-theoretically secure
  - No amount of NSA tomfoolery can weaken this...
- Difficult to identify "related" shares on different archives
  - Several approaches to make this possible

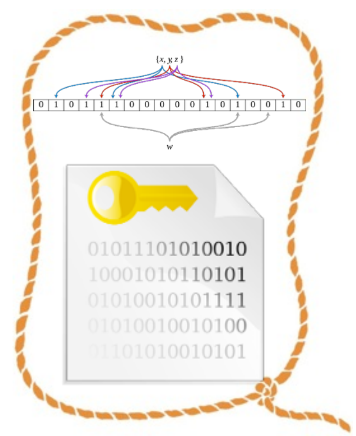
Challenge 2: search the data

- This level of security is great, but...
- How can we ***find*** anything in this system?
  - Want to prevent archive maintainers from figuring out what we're looking for
  - Want to prevent archive maintainers from identifying relationships between shares
- Client needs to tag shares on each archive
  - Tags need to be "nonsense" to archive
  - Tags need to be different across archives
  - Need to prevent (or at least reduce) possibility of correlating documents by monitoring search requests
  - But, tags need to be readily searchable (of course)

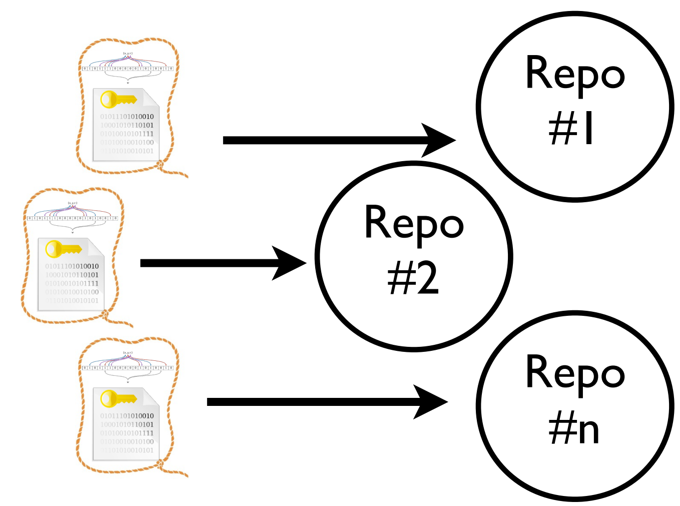
Percival overview

File Ingestion

For each file

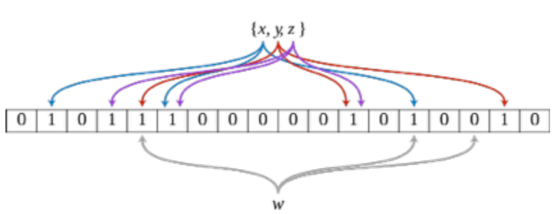


Generate a Bloom filter
for each share



Distribute these
bundles, one per
repository

Searching



Create a Bloom filter
from the search
terms

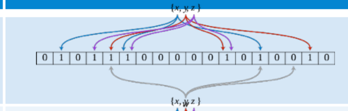
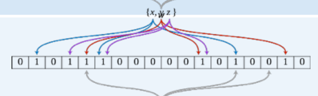
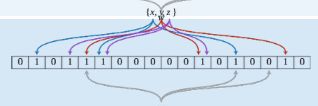
Client Side

Server Side

Compare it to each
share's filter, and
generate results map

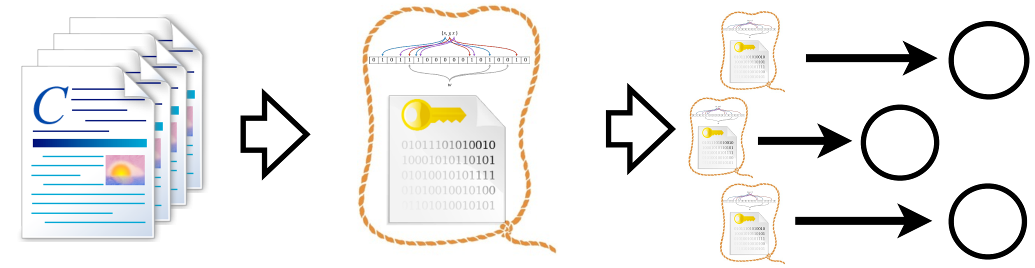
Server Side

Client Side

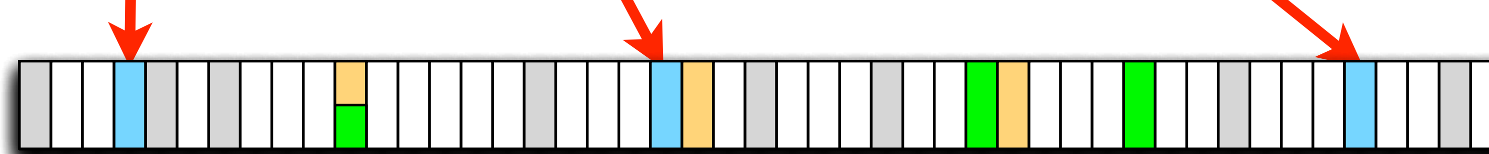
filename	results filter
file1	
file2	
file3	

Process the results

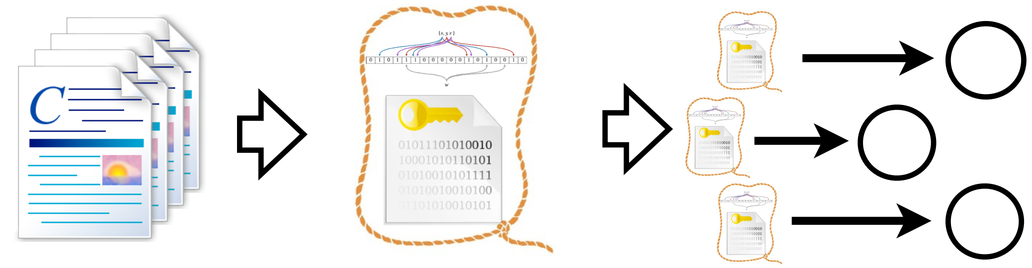
Design: ingestion



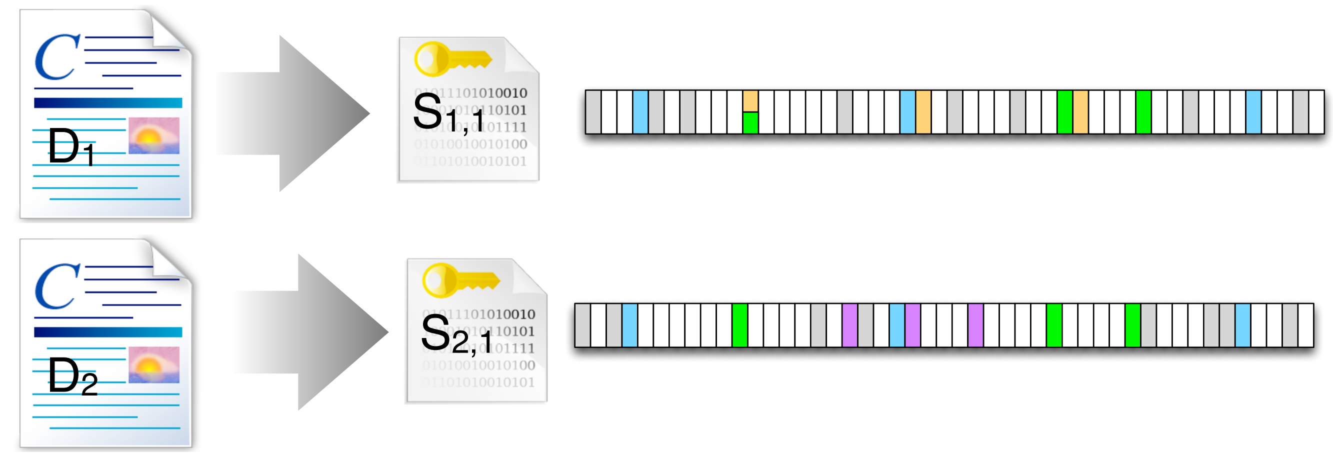
- Pre-index each share with a Bloom filter
  - Generate list of terms *W*
  - Combine each term, w_i, with the repository key, key_r
    $v_i = KeyedHash(w_i, key_r)$
  - Generate *k* locations using *k* hash functions of v_i and set the corresponding bits in the Bloom filter for *r*
- Problem: it may be possible to associate shares on *r* with the same bits set in the Bloom filter
- Solution: set randomly-selected bits in the Bloom filter for each share on each repository (chaff)
  - Obscures the relationship between set bits and terms
  - Increases the number of false positives



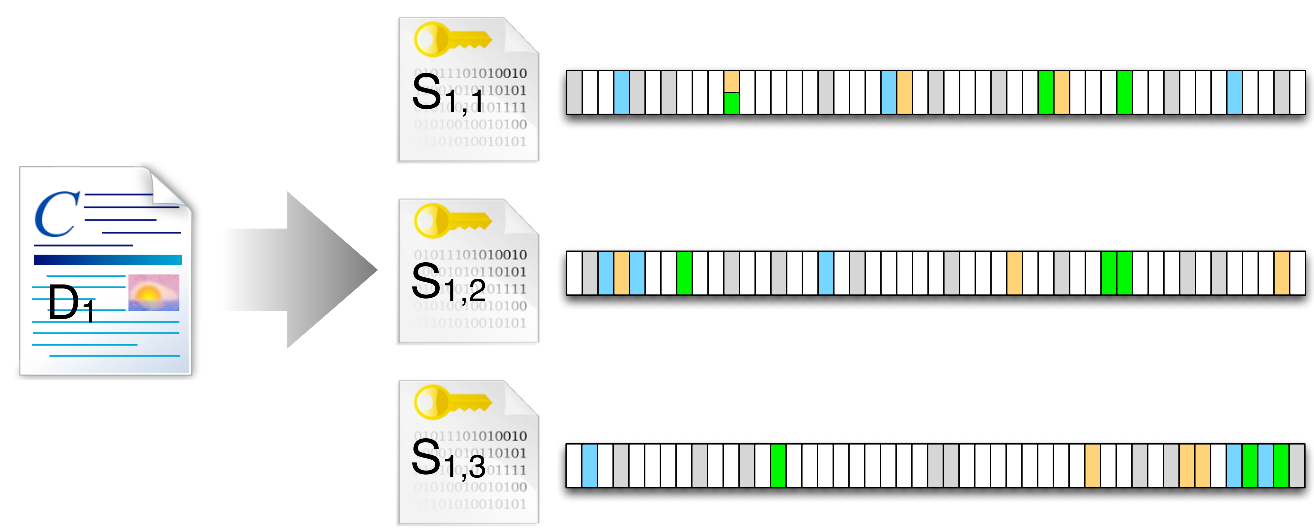
Design: ingestion



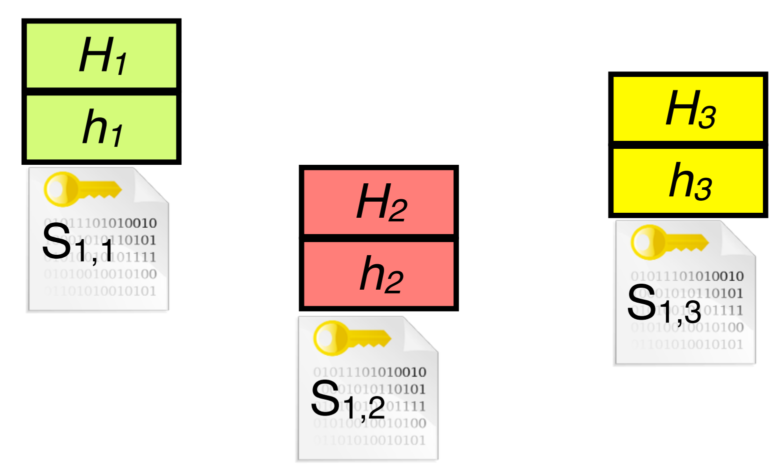
- Shares with similar terms still differ in Bloom filters
  - Amount of chaff is tunable—currently investigating tradeoffs



- Different Bloom filter for each repository
  - Difficult to correlate shares across repositories



- Add H_i, h_i to each share
  - H = hash(*data*)
  - H_i = hash (H, key_r)
  - Share of H: h_i = split (H, i)



Design: search

Client

Server

Client:

- Generate a search Bloom filter for each repository
- Send each Bloom filter and hit threshold to each repository

Server:

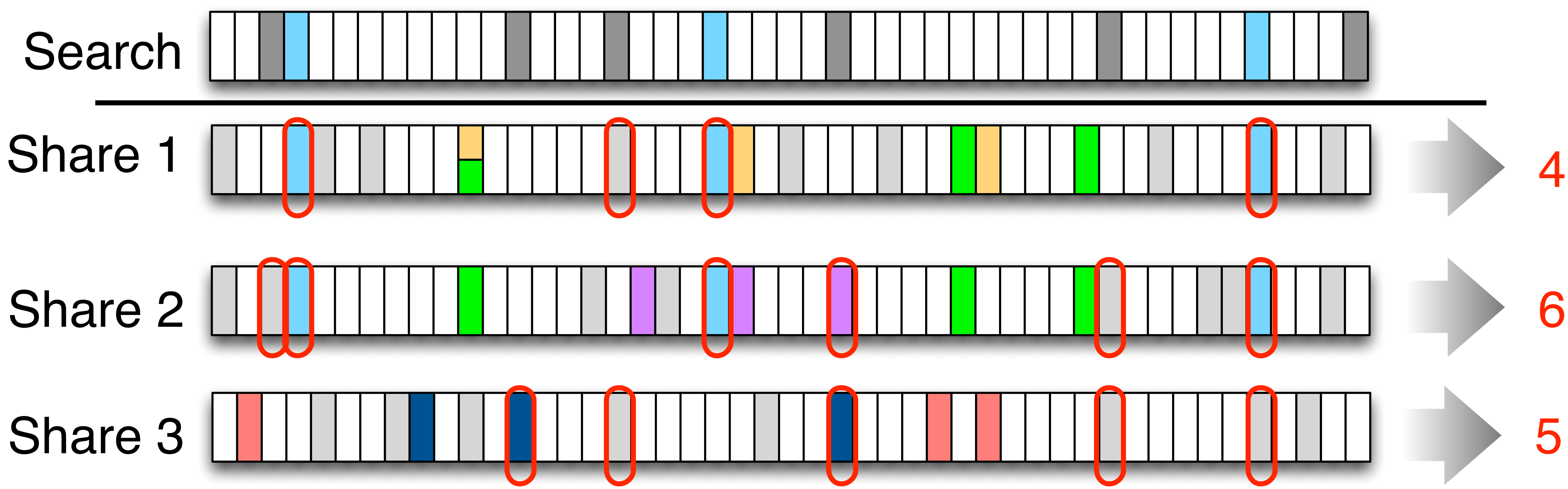
- Calculate intersection for each share's Bloom filter
- Hit threshold met?
- Return list of shares that meet the threshold

Client:

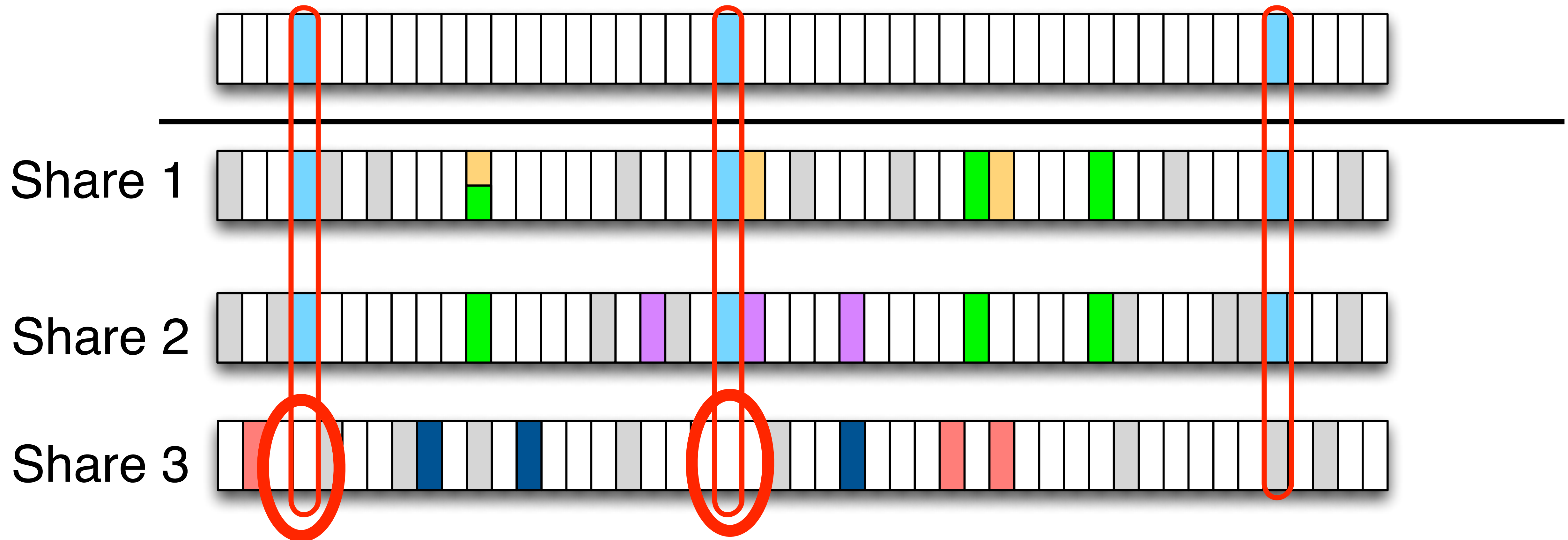
- Get results from each server
- Identify documents with shares in each result list
- Request shares from each repository

Search: using the Bloom filters

- Set *b* bits in search Bloom filter using same hash functions that were used when shares were stored
  - Use key_r to generate different filters for each repository
- Add chaff bits to search Bloom filter
  - Again, goal is to make correlating different searches more difficult
- Require archive to return all results with at least *b* bits that match
  - This contains a superset of desired results



Search: identifying results at the client



- Eliminate shares whose Bloom filters don't contain all of the "real" bits
- Try all combinations of shares, one from each repo
  - Reassemble the hash value from the split hashes
  - Verify reassembled value using *key*$_r$ against keyed hash stored in one of the shares
- Request full shares to rebuild the desired data

Search: issues

- Is combinatoric reassembly slow?
  - Depends on the number of shares that pass the Bloom filter test
  - Typically not an issue with low false positive rates
  - Can become large for large share "width"
- Is use of Bloom filters slow or inefficient?
  - Can use techniques for faster searches
  - Can compress Bloom filters (especially results)
    - Results need only include bits that match the search

How secure is it?

- Data can't be rebuilt without sufficient shares
  - Attempts to get large quantities of data from independent archives will raise suspicion
- What about targeted attacks?
  - Difficult to correlate searches across archives to identify related shares
  - Recombination is much harder without eliminating shares that don't contain all search term bits
- Can attacker learn search terms?
  - Set bits are different for each archive
  - Set bits are obscured in both index and search filters

- Currently investigating *how* well this hides information...

Where are we now?

- Working on a prototype with Sandia National Labs
- Investigating tradeoffs in
  - Obfuscation of bit groups
    - Adjust filter size → loading → false hit rate
  - Methods to mitigate false hit rate
  - Methods to increase computational bounds to determine key_r
- Exploring long-term attacks that attempt to correlate searches, even with chaff on both ingest and search
- Working on better ways to split secrets more efficiently
- Rebuilding shares after an archive failure

Preserving the meaning of digital data

- Digital data may not have an obvious meaning
- Some digital data is (relatively) simple to interpret
  - ASCII text
  - GIF (only a 33 page standard!)
    - Other image types are more complex
- Other data is more difficult to interpret
  - Microsoft Word & Excel
  - PostScript / PDF: interpreted languages!
  - Databases are very difficult to deal with
- Standards change over time: how can old documents be read today?

Preserving meaning: emulation

- One solution is to keep a virtual execution environment that knows how to interpret a document
  - Use provenance to track which environment is necessary
  - Share environments wherever possible
    - Example: single environment for MS Word 97
    - Need to be aware of "implicit" customizations!
- Problem: keeping execution environments running isn't so easy
  - Use simplified environments (fancy Turing machine)
  - Layer more complex environments on top of one another
  - This approach may be slow

Preserving meaning: migration

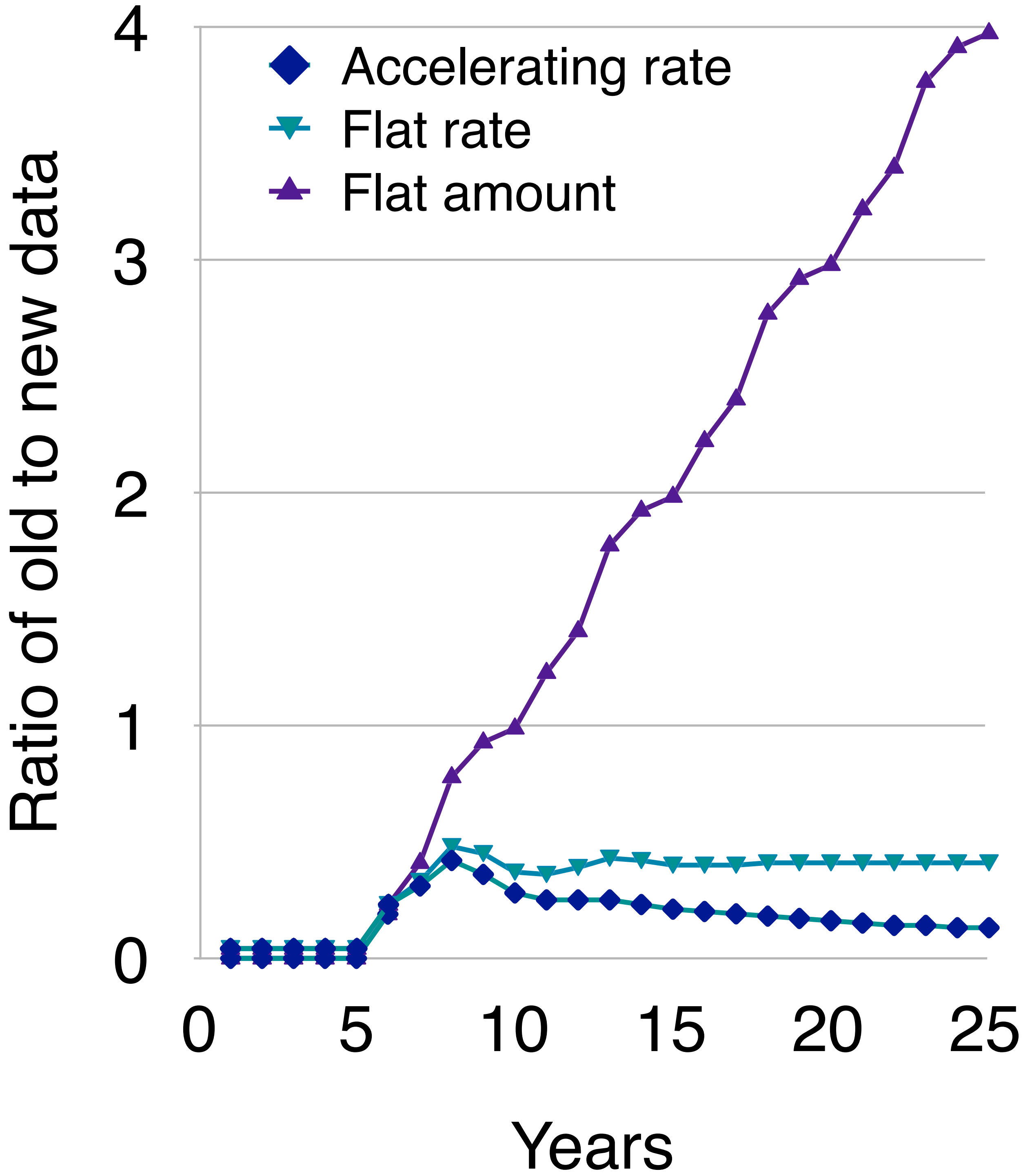
- Alternate solution: refresh representation on copy
  - Can be done as part of copy to new devices
  - May keep the original version around, just in case
- Benefits
  - No need to keep a complex virtual environment available
  - There's always software to read the most recent version
- Drawbacks
  - Translated copy may not have all the functionality of the original
    - Example: PDF rendered to a bitmap image
  - Does the translated copy *really* have the same meaning as the original?

The economics of archival storage

- Users want to pay for archival storage once: when data is created
  - New data is most frequently used
  - Many models collect money from usage (Flickr, YouTube)
- Problem: archival storage has ongoing costs!
  - Refresh cycles for data and media
  - Management costs
- **Usage falls off dramatically as data ages!**
  - Trade off high initial cost against high ongoing costs?
    - Fewer refresh cycles & lower management cost?
  - Pay for ongoing storage with revenue from new data?
    - Depends on increasing growth rate: not sustainable in the long term
  - Get rid of much of the data
    - Which data and who decides?

So where are my photos?

- Exponential growth in demand for first 5 years
  - Slows a bit in years 4–5
- Increasing growth rate
  - New storage costs dominate existing storage
  - Ratio of old:new drops over time
- Level growth rate
  - Old data : new data ratio remains approximately constant
- Level growth amount
  - Old data dominates quickly

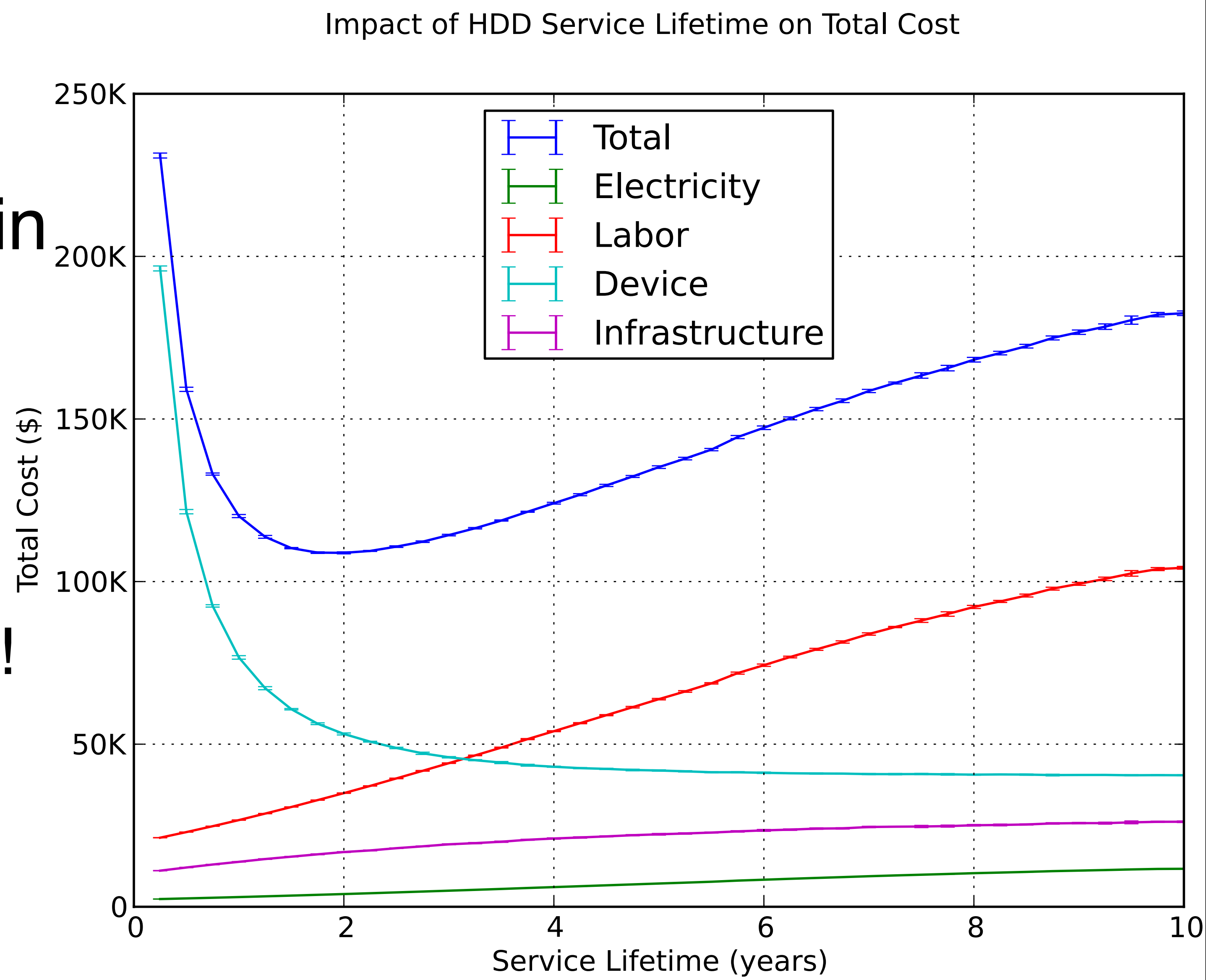


How can we predict long-term storage costs?

- Build a model that incorporates
  - Predicted storage costs and density
  - Models of storage reliability
  - “Cost of money”: buy things now or later?
  - *All of these may vary over time...*
- Model must include
  - “Predictable” costs
  - Random events that impact cost
- Use Monte Carlo simulation
  - Run the model hundreds of times
    - Need multiple runs to capture impact of random events
  - Use different assumptions for some sets of runs

Q: When should we replace storage with a "better" model?

- Build archive from disks
  - Capacity grows over time: doubles each year
- How long should disks remain in service?
  - Until they die?
- **Conclusions**: replace after about 2–3 years
  - Even if disks could last longer!
  - Depends on capacity growth rate over time
- May not hold true
  - As disk growth rate slows
  - If we use NVRAM instead of disk



Ongoing research

- Study trade-offs between endowment size, data protection level, and archive survivability
- Study real-world scenarios/events:
  - Compare various storage media (disk, flash, cloud, etc.) for suitability in archival storage
    - Trade off longer lifetime for higher up-front cost?
    - Focus on higher reliability or higher density / lower cost?
  - Experiment with various data and media capacity growth rates
  - Examine impact of disruptive technologies

Is a digital Dark Age coming?

- Many users keep their "digital lives" online
  - Personal communications
  - Photos & video
- Sites typically supported by ads
  - Users look at "new stuff" a lot
  - Older stuff is rarely accessed: no opportunity to sell ads!
- What's going to happen to 5–10 year old photos?
  - Old data will dominate capacity and cost
  - Companies may start to prune cold data, like old photos
  - Will you notice? Will you care?
- Are you willing to pay for long-term archiving?
  - Chronicle of Life Foundation...

Summary: challenges in preserving data for the long-term

- Archiving 10^{18} bits
  - Reliability, integrity & security
  - Indexing and searching
  - Scalability
  - Management
- Ensuring the bits can be used in decades
  - Migration
  - Emulation
- Integrating all of the solutions into a system that can survive for a century or more
  - This is a *very* difficult challenge
  - Involves issues of economics and policy as well as technology

Conclusions

- Long-term digital data preservation is critically important
  - The fate of the world's data is at stake!
  - The problem isn't going away
  - The problem is getting worse ... fast!

- Data preservation is largely *ad hoc* today
  - There are solutions, but they address only one or two issues (at most)
- Many problems are left to be solved
  - Research has high potential for impact

Other research areas

- Scalable file systems
  - Ceph: highly scalable file storage for HPC
  - Algorithmic distribution of data for scalability
  - Security: authorization and protection for data at rest
  - Search
- Non-volatile memories
  - Integrating object storage and NVRAM
  - Data layout and wear leveling for byte-addressable NVRAM
- Shingled disk: layout and management techniques

- Collaboration with industry: Pure Storage
  - Many other project-level collaborations

Questions?

http://www.ssrc.ucsc.edu/proj/archive.html

Collaborators (partial list)

- Ian Adams
- Joel Frank
- Kevin Greenan
- Thomas Kroeger
- Darrell Long
- Brian Madden
- Daniel Rosenthal
- David Rosenthal (no relation)
- Thomas Schwarz
- Mark Storer
- Kaladhar Voruganti
- Avani Wildani
- Erez Zadok