Saving Bits Forever:
A Systems View of Long-term Digital Storage

Mary Baker
HP Labs
http://www.hpl.hp.com/personal/Mary_Baker

Outline

- The digital preservation problem
- Strategies for addressing the problem
- LOCKSS (Lots of Copies Keeps Stuff Safe)
- The Pharaoh Project
- Conclusion:
  - Static data requires dynamic system

The need for long-term digital storage

- Emerging web services
  - Email, photo sharing, videos, web site archives
- Regulatory compliance and legal issues
  - Sarbanes-Oxley, HIPAA, intellectual property litigation
- Many other fixed-content repositories
  - Scientific data, intelligence info, libraries, movies, music

YAHOO! MAIL

Gmail +talk BETA

snapfish

One Hundred Seventh Congress of the United States of America

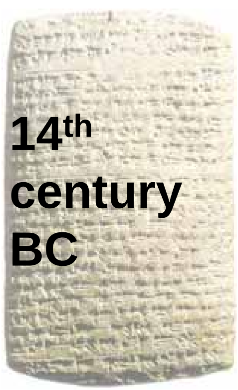
AT THE SECOND SESSION

The Absinthe Literary Review

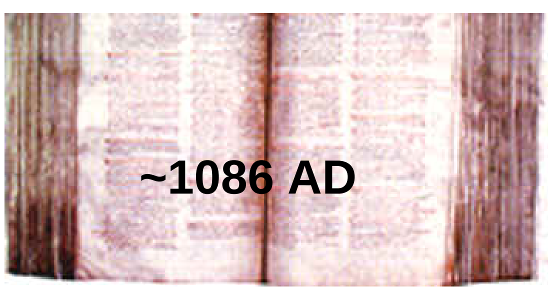
SUMMER/AUTUMN 2005

Physical to virtual transformation

- Tools to move from analog to digital content
  - But no understanding of how to keep digital content
- We're used to throwing technology away
  - But now we have assets beyond the technology
- We've created an explosion in fixed content
  - Some of which we may want to keep forever



14th century BC



~1086 AD

1970's

Why is long-term storage hard?

- Large-scale disaster
- Human error
- Media faults

Long-term content suffers from more threats than short-term

- Component faults
- Economic faults
- Attack
- Organizational faults

- Media/hardware obsolescence
- Software/format obsolescence
- Lost context/metadata

Why is long-term storage hard?

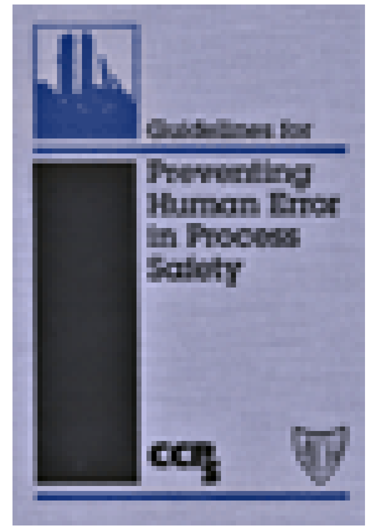
- Large-scale disaster
- Human error
- Media faults

- Component faults
- Economic faults
- Attack
- Organizational faults

- Media/hardware obsolescence
- Software/format obsolescence
- Lost context/metadata

Why is long-term storage hard?

- Large-scale disaster
- Human error
- Media faults



- Component faults
- Economic faults
- Attack
- Organizational faults

- Media/hardware obsolescence
- Software/format obsolescence
- Lost context/metadata

Why is long-term storage hard?

- Large-scale disaster
- Human error
- Media faults
- Component faults
- Economic faults
- Attack
- Organizational faults
- Media/hardware obsolescence
- Software/format obsolescence
- Lost context/metadata

Why is long-term storage hard?

- Large-scale disaster
- Human error
- Media faults

- Component faults
- Economic faults
- Attack
- Organizational faults

- Media/hardware obsolescence
- Software/format obsolescence
- Lost context/metadata

Why is long-term storage hard?

- Large-scale disaster
- Human error
- Media faults

- Component faults
- Economic faults
- Attack
- Organizational faults

- Media/hardware obsolescence
- Software/format obsolescence
- Lost context/metadata

Why is long-term storage hard?

- Large-scale disaster
- Human error
- Media faults

- Component faults
- Economic faults
- Attack
- Organizational faults

- Media/hardware obsolescence
- Software/format obsolescence
- Lost context/metadata

Why is long-term storage hard?

- Large-scale disaster
- Human error
- Media faults

- Component faults
- Economic faults
- Attack
- Organizational faults

- Media/hardware obsolescence
- Software/format obsolescence
- Lost context/metadata

Why is long-term storage hard?

- Large-scale disaste
- Human error
- Media faults

- Component faults
- Economic faults
- Attack
- Organizational faults

- Media/hardware obsolescence
- Software/format obsolescence
- Lost context/metadata

Why is long-term storage hard?

- Large-scale disaster
- Human error
- Media faults

- Component faults
- Economic faults
- Attack
- Organizational faults

- Media/hardware obsolescence
- Software/format obsolescence
- Lost context/metadata

Why is long-term storage hard?

- Large-scale disaster
- Human error
- Media faults

- Component faults
- Economic faults
- Attack
- Organizational faults

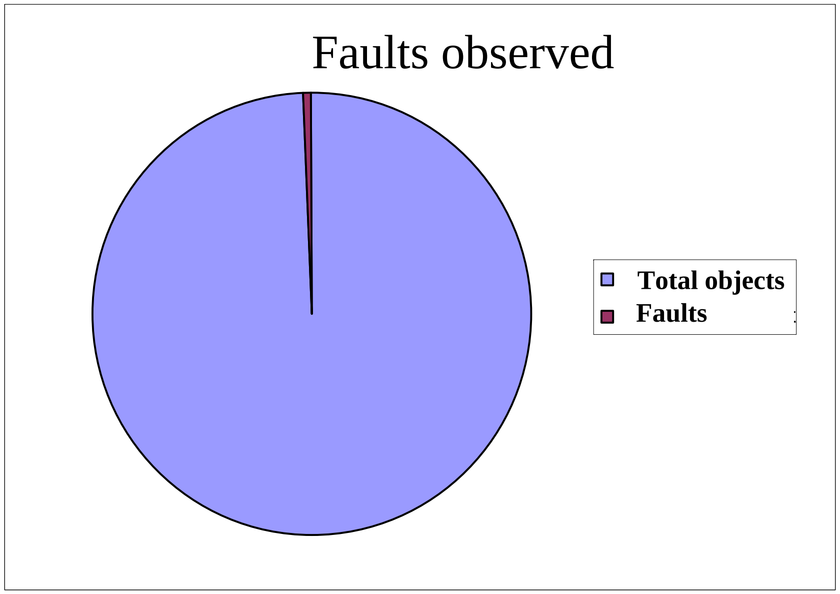
- Media/hardware obsolescence
- Software/format obsolescence
- Lost context/metadata

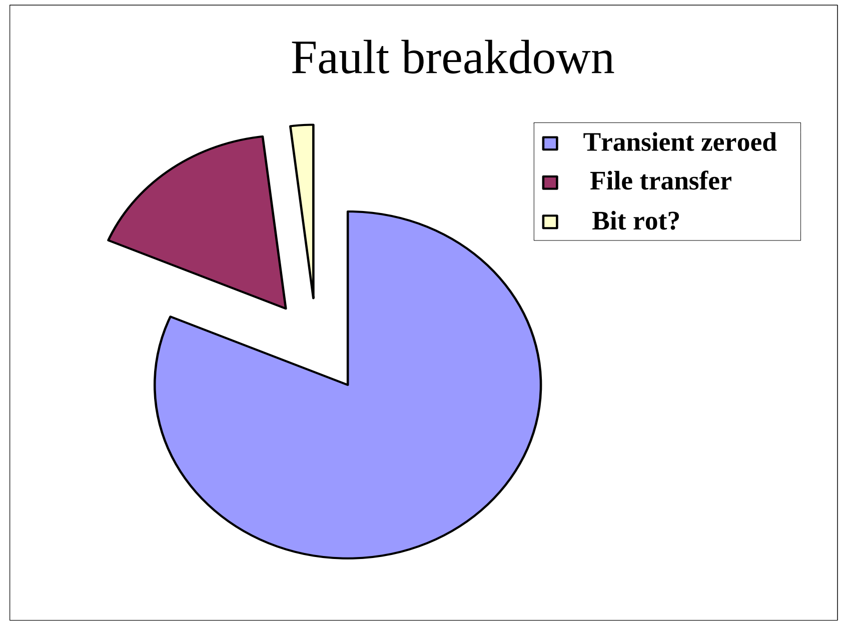
Why is all this still a problem?

- Assumption of sufficient budget
- Assumption of replica independence
- Assumption of fault visibility, but latent faults…
  - Lurk subversively until data accessed
  - Aren’t unearthed through archival workloads
  - Accrue over time until too late to fix
  - Become significant
    - At large scale
    - Over long time periods

Latent and transient faults on disk

Source: 810 days of Internet Archive failure data over 1896 disks





Total: 1/ 234

Total objects: 1,492,993
Total errors: 6412

Transient: Zero read: 5046 = 1 / 296

Latent: File transfer: 1218 = 1/ 1226

Latent: Bit Rot?: 148 = 1 / 10088

Strategies for dealing with this mess

- Address high costs of preservation
  - Commodity hardware
  - Reduce on-going costs
  - Better cost models
- Replicate content, break correlations between replicas
  - Geographic, administrative, platform, media, formats…
- Audit replicas proactively to detect damage
  - Data must be accessible to do this cheaply!
- Migrate content to maintain usability
  - To new hardware, formats, keys…
- Avoid external dependencies
  - Includes vendor lock-in, DRM issues
- Plan for data exit

The LOCKSS solution:
Exploit natural replication across libraries

"… let us save what remains: not by vaults and locks which fence them from the public eye and use in consigning them to the waste of time, but by such a multiplication of copies, as shall place them beyond the reach of accident."

Thomas Jefferson, 1791

Exploit existing replication

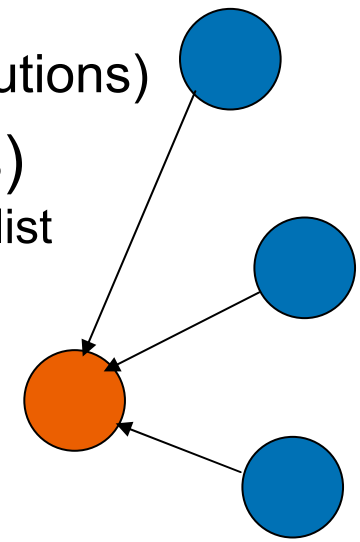
- Lots of Copies Keep Stuff Safe www.lockss.org
- Many libraries subscribe to the same materials
- Appliance used by libraries around the world
  - Cheap PC with some storage
  - Libraries maintain existing relationships with publishers
  - Materials subscribed to collected/preserved by LOCKSS
  - Run a P2P audit/repair protocol between LOCKSS peers
  - Not a file sharing application!
- Survive or degrade gracefully in the face of
  - Latent storage faults & sustained attacks
- Make it hard to change consensus of population

The LOCKSS audit/repair protocol

- A peer periodically audits its own content
  - To check its integrity
  - Calls an opinion poll on its content every 3 months
  - Gathers repairs from peers
- Raises alarm when it suspects an attack
  - Correlated failures
  - IP address spoofing
  - System slowdown
- Currently updating deployed system

Sampled opinion poll

- Each peer holds for each document
  - *Reference list* of peers it has discovered
  - *History* of interactions with others (balance of contributions)
- Periodically (faster than rate of storage failures)
  - *Poller* takes a random sample of the peers in its reference list
  - Invites them to *vote*: send a hash of their replica
- Compares votes with its local copy
  - Overwhelming agreement (>70%) ☞ Sleep blissfully
  - Overwhelming disagreement (<30%) ☞ Repair
  - Too close to call ☞ Raise an alarm
- Repair: peer gets pieces of replica from disagreeing peers
  - Re-evaluates the same votes
- Every peer is both poller and voter



Opportunities that make this possible

- Massive redundancy ``for free''
  - Peers (libraries) demand whole local replicas of content
  - Replicas independent of each other
    - Geographic, administrative, platform, technology, financially…
- Digital preservation is about preventing change
  - Not precipitating it
  - Efficient system is *not* a goal
  - Go no faster than necessary to fail as slowly as possible

Threat model

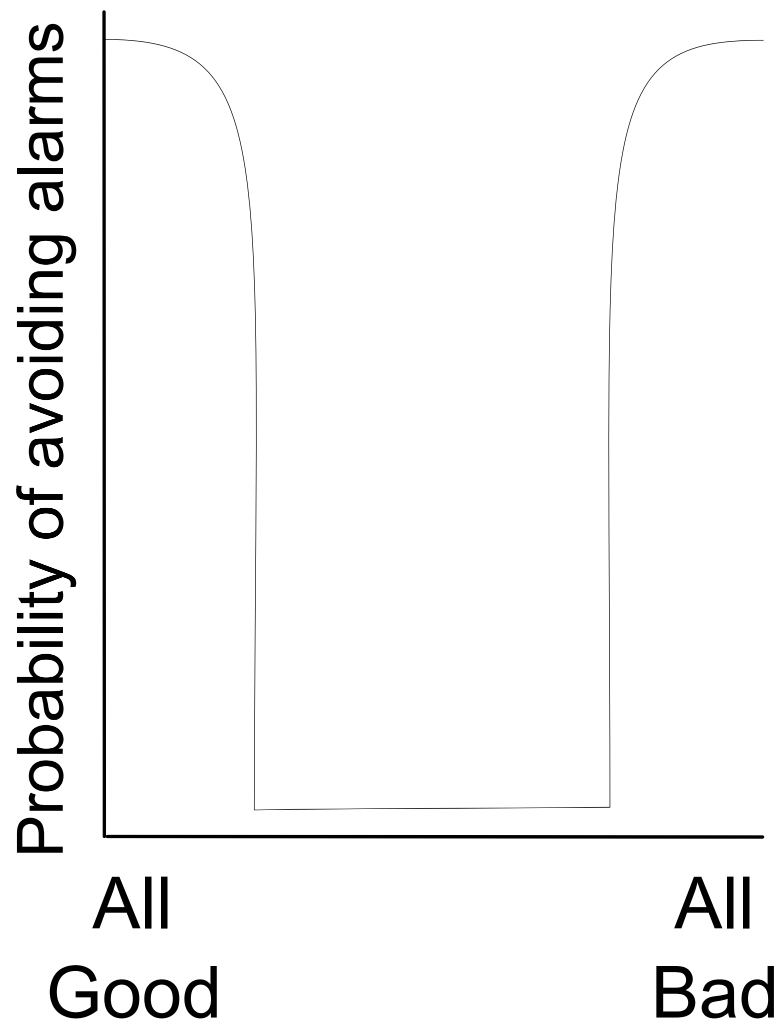
- Beyond natural damage, assume we're attacked
  - Platform/social attacks
- Mitigate further damage through protocol
- Top adversary goals
  - Stealth modification
    - Modify majority of replicas to contain adversary's version
    - Without getting caught (setting off alarms)
  - Attrition (denial of service)
    - Waste peers' resources at network, application, human layers
    - Prevent audit until storage failures overwhelm & damage system
- Other adversary goals
  - Content theft, free-riding, false alarms, etc.

Limit the rate of operation

- During initiation of new polls
  - Peers determine their rate of calling polls autonomously
  - No changes due to external stimuli
  - Adversary must wait for next poll to attack as a voter
- Keep poll rate constant to cap attack rate
- Go no faster than necessary
  - So system fails as slowly as possible

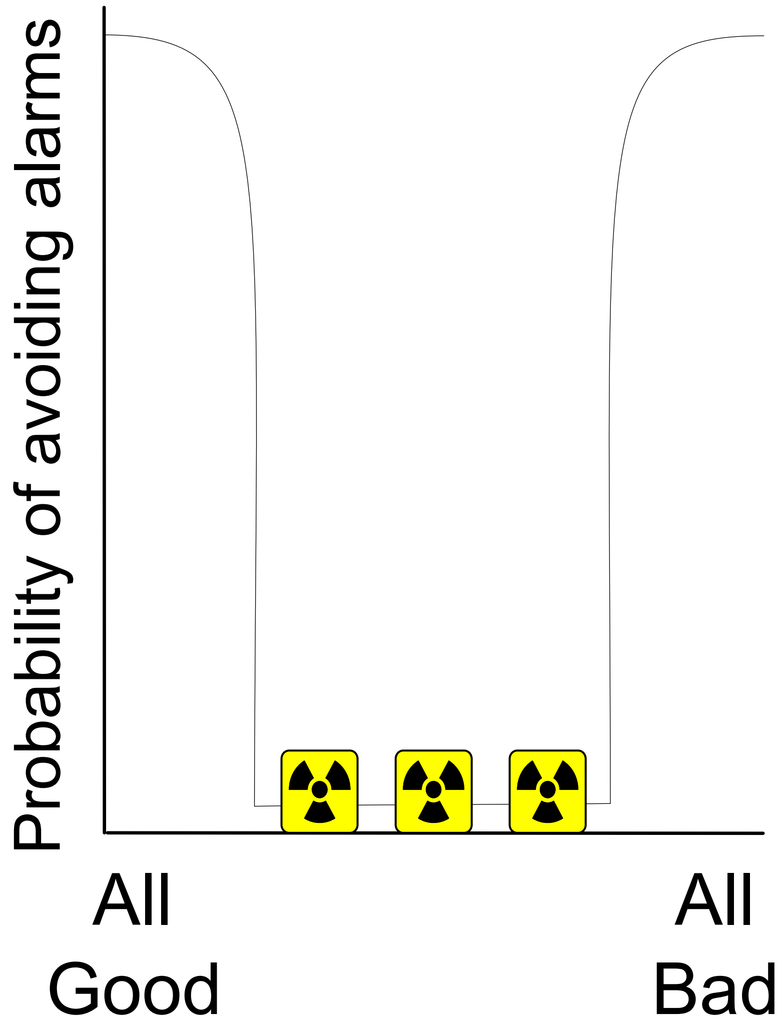
Bimodal alarm behavior

- Most replicas the same
  - No alarms



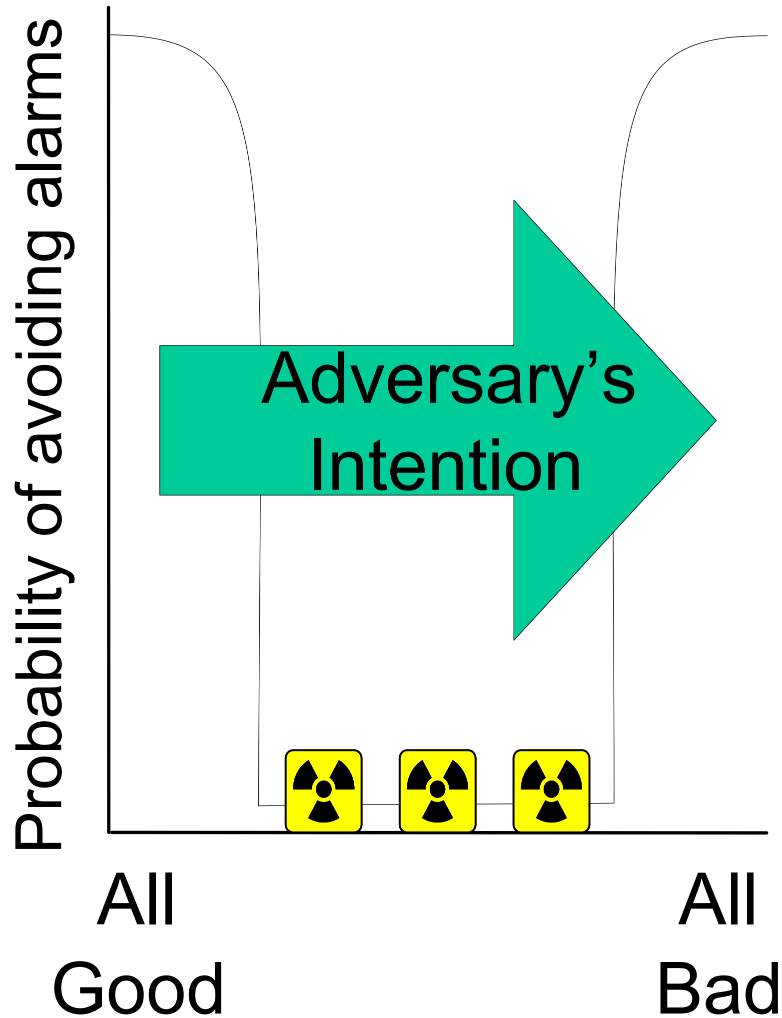
Bimodal alarm behavior

- Most replicas the same
  - No alarms
- In between
  - Alarms very likely

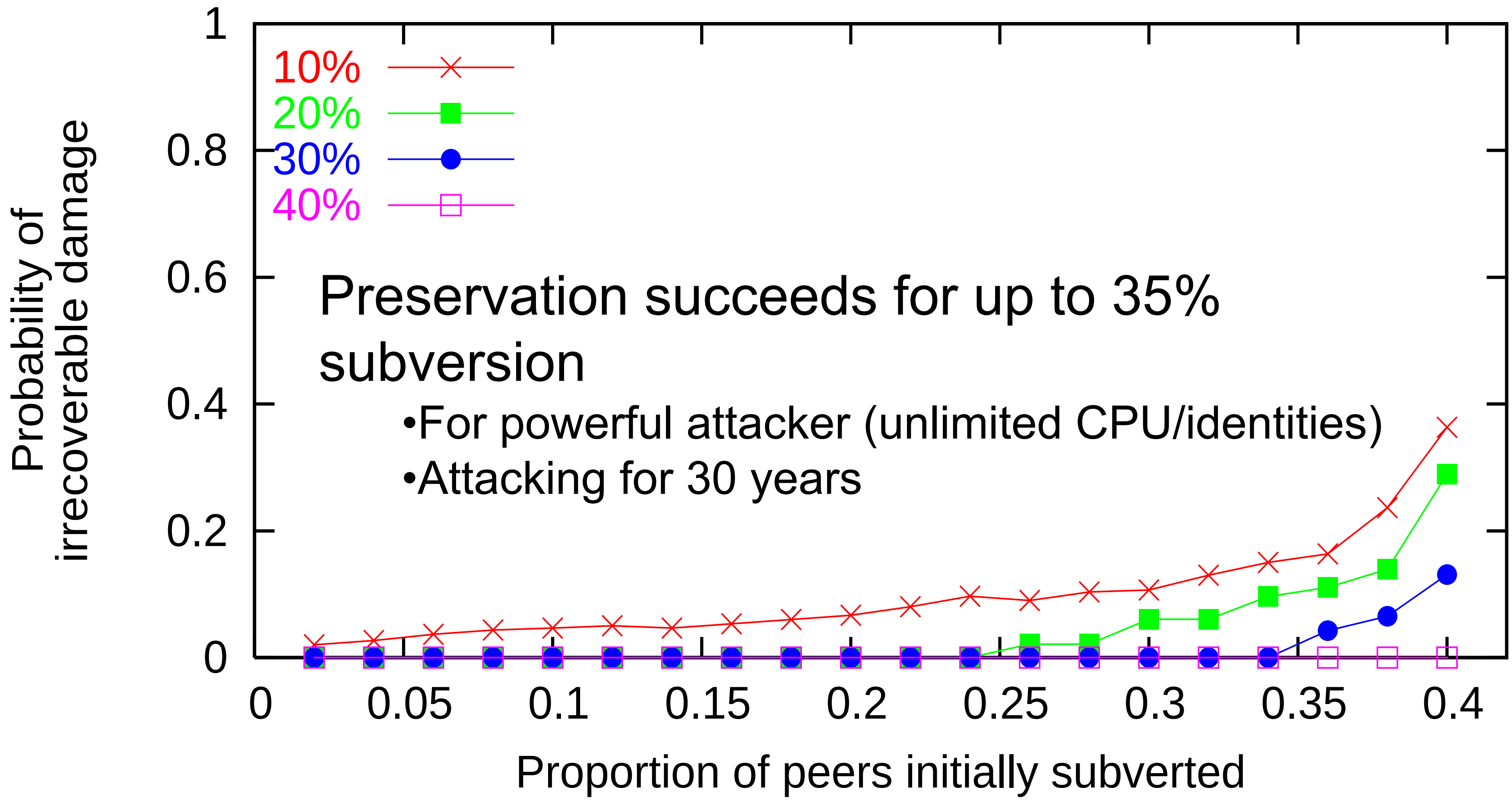


Bimodal alarm behavior

- Most replicas the same
  - No alarms
- In between
  - Alarms very likely
- To achieve corruption
  - Adversary must pass through “moat” of alarming states
  - Damaged peers vote with undamaged peers
  - Rate limitation helps



Probability of irrecoverable damage



The Pharaoh Project

Low-cost, long-term, reliable storage for large repositories

The problem with large repositories

- Few replicas naturally available
  - Initial storage outlay can be daunting
- Large on-going costs
  - Data center space is expensive
- Preservation “best practices” contradict IT trends
  - Consolidation versus replication
  - Homogeneity versus diversity
  - Administrative centralization versus independence

Why do we have a chance?

- Exploit replication for disaster recovery
  - No longer require backup processes
  - Poor synchronization between replicas okay
- Repository workload has limited requirements
  - Does not need low latency access
  - Does not need high rate of update in place
- Use commodity storage to bring down outlay costs
  - Address reliability with audit processes
  - Use an easily evolvable architecture
- Bring down on-going costs through spin-down, etc.

How do we evaluate trade-offs?

- How much replication?
- How reliable do individual replicas need to be?
- How do we audit?
  - What?
  - Where?
  - How often?
- Latency/power/reliability?
- We need better modeling tools!

Can we model long-term reliability?

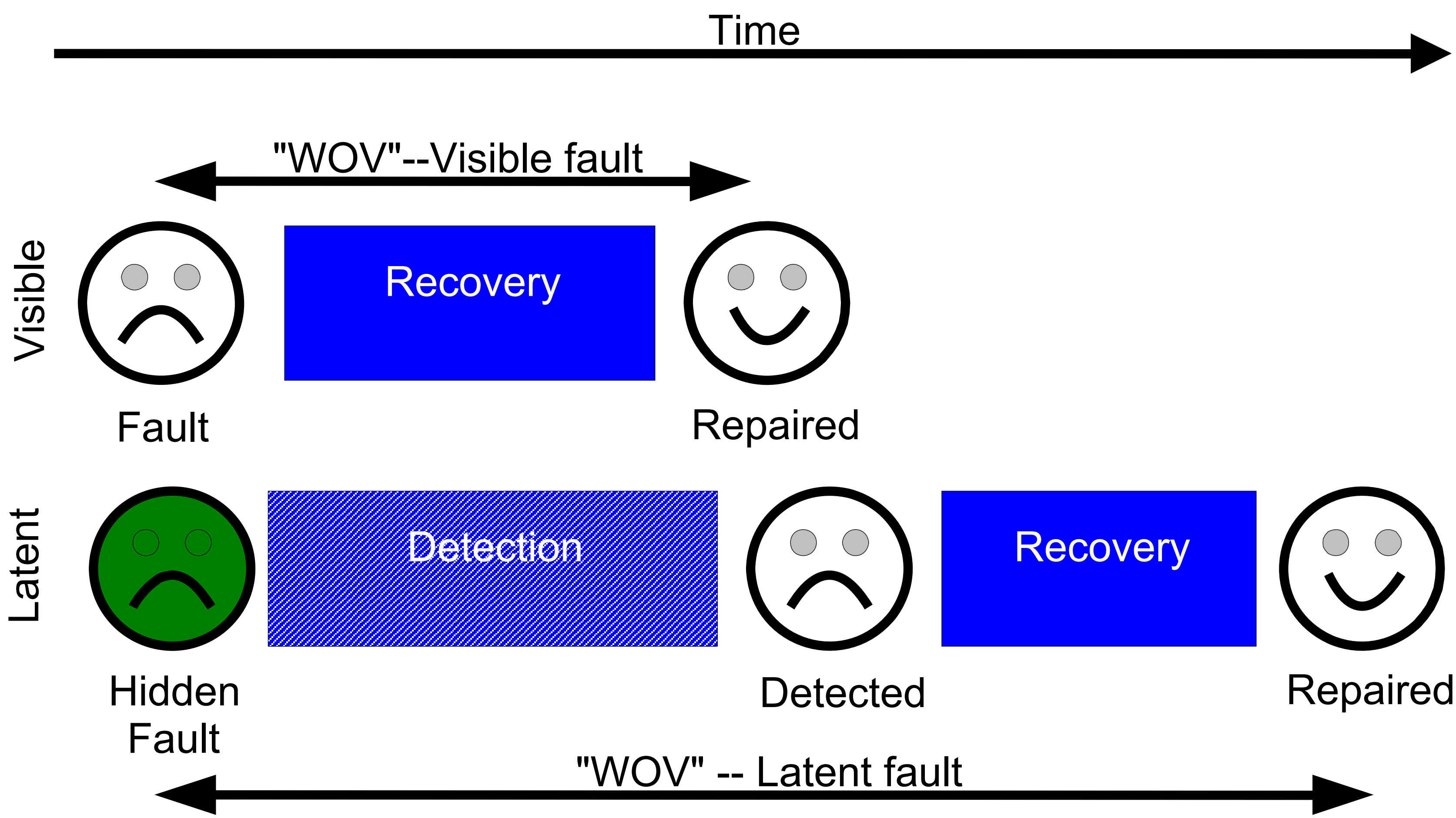
- Abstract reliability model for replicated data
  - Applies to all units of replication
  - Applies to many types of faults
- Extend RAID model
  - Account for latent as well as visible faults
  - Account for correlated faults: temporal and spatial
- Simple, coarse model
  - Suggest and compare strategies (choose trade-offs)
  - Point out areas where we need to gather data
- *Not for exact reliability numbers*

Our current approach

- Start with two replicas, then add more
- Derive MTTDL of mirrored data in the face of
  - Both immediately visible and latent faults
- Mirrored data is unrecoverable
  - If copy fails before initial fault can be repaired
- Time between fault and its repair is
  - *Window of Vulnerability* (WOV)

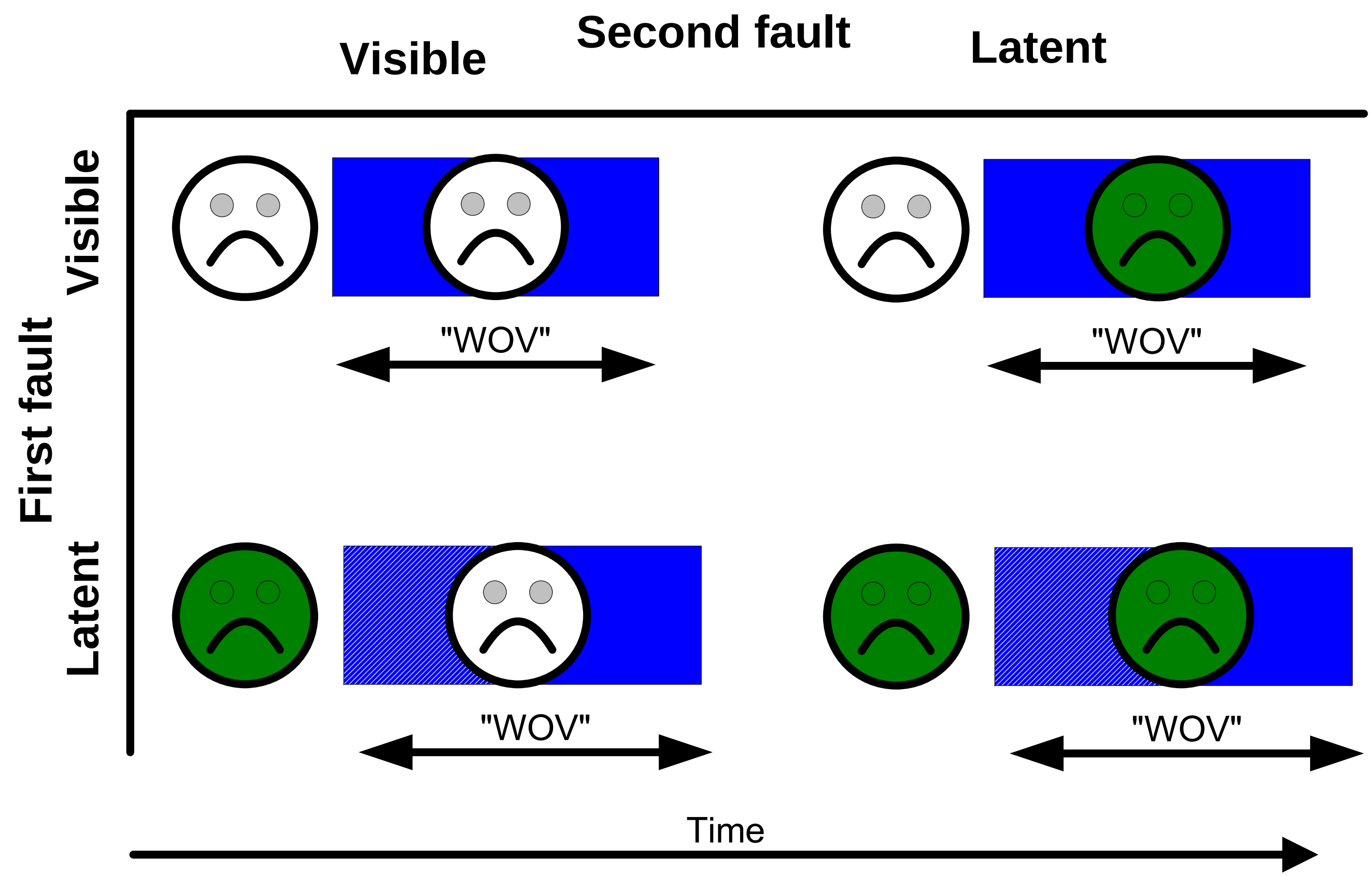
Window of vulnerability

Temporal overlap of faults



- Want detection time to be small

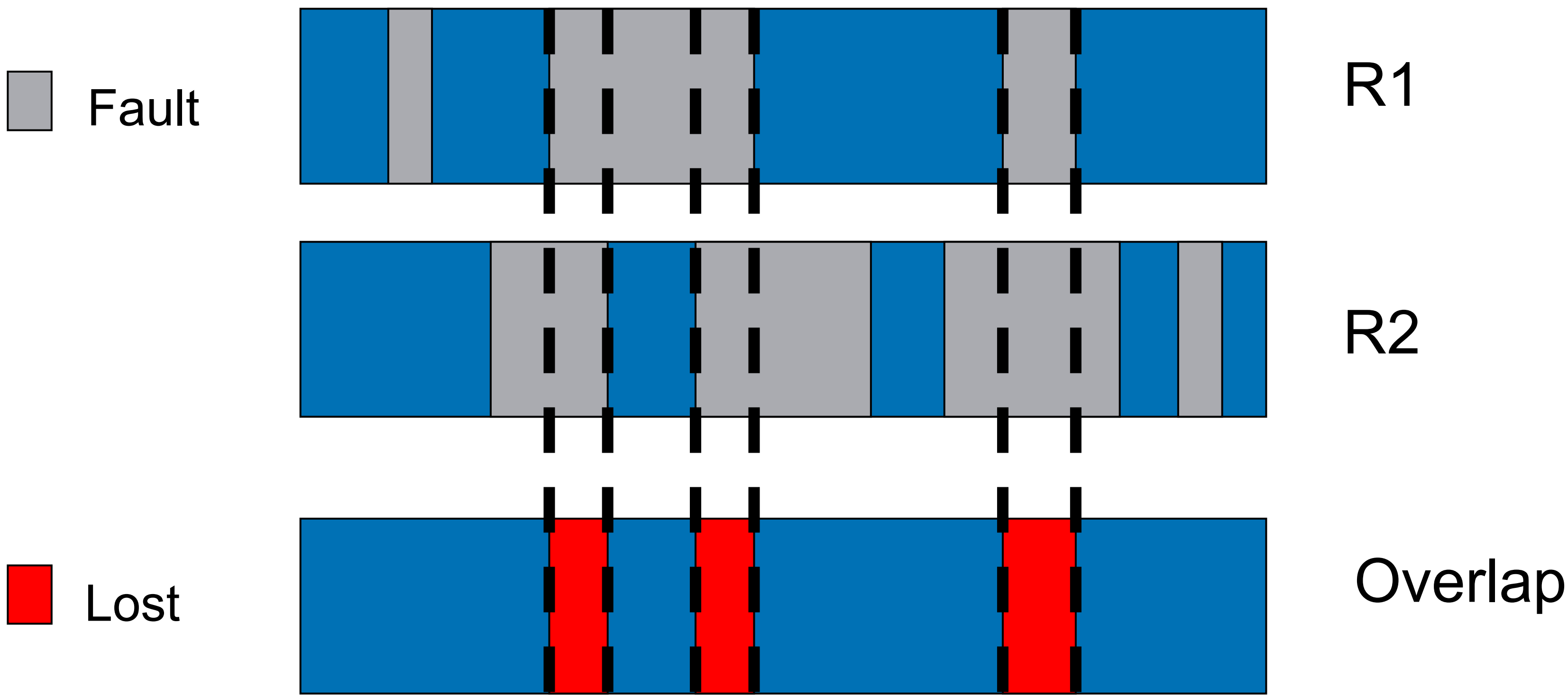
Data loss cases with 2 replicas



- Overall probability = sum of each case

Spatial overlap of faults

- Temporal overlap alone overstates likelihood of data loss



- Faults may be bits, sectors, files, disks, arrays, etc.
- If any two faults overlap, data is lost
- The smaller the faults, the less likelihood of

Completing the model

- Multiply temporal and spatial probabilities
  - For each of the four loss cases
- Correlation: use multiplicative scaling factors for
  - Temporal correlation of faults
  - Spatial correlation of faults
- We also extend the model for further replication

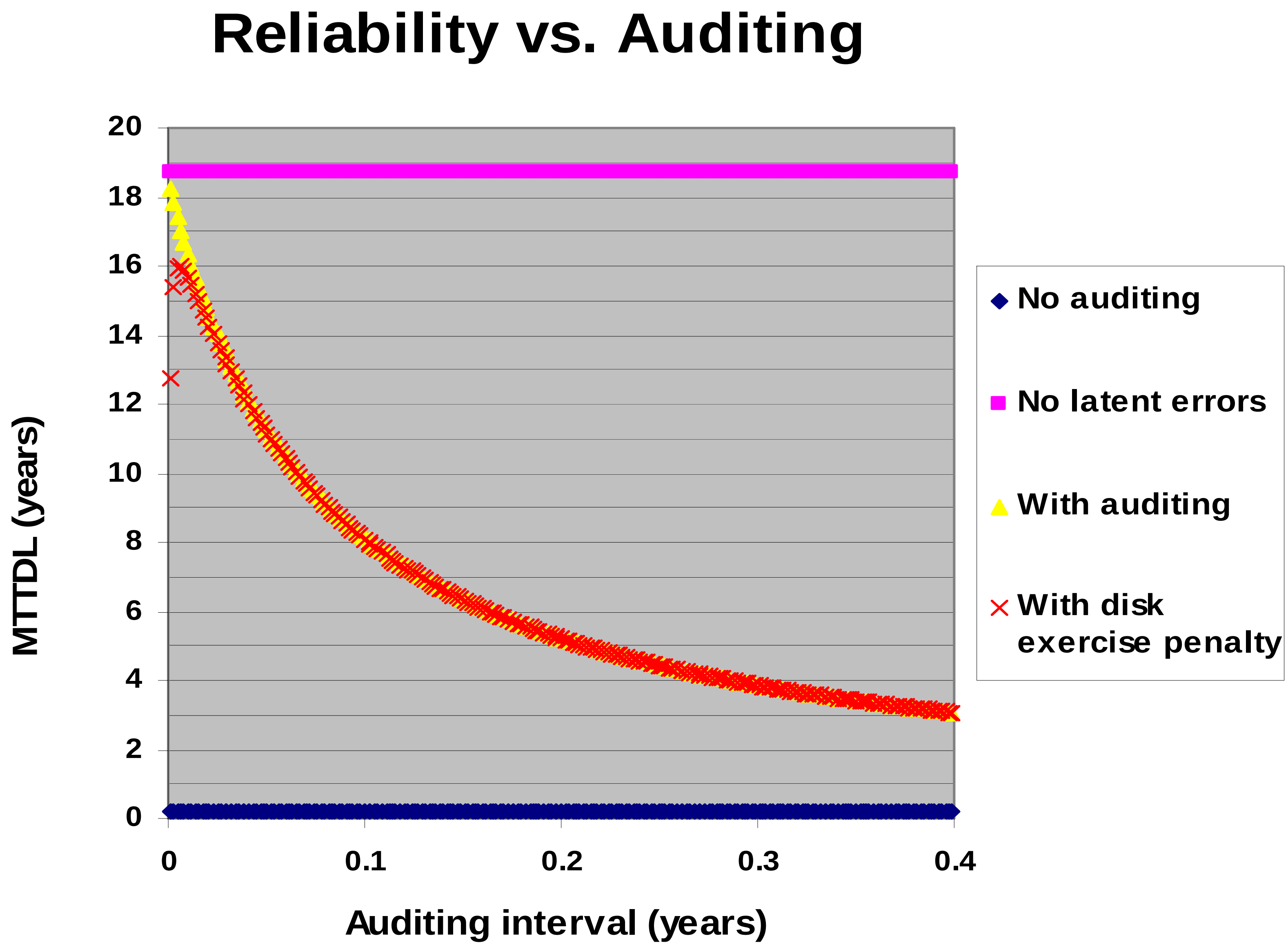
Implications

- Must audit for latent faults
  - Important to reduce detection time
  - Even if latent faults are infrequent
  - Content must be accessible to do this cheaply!!
- Need independence of additional replicas
- MTTDL varies quadratically with both MV & ML
  - Cannot sacrifice one for the other
- If sizes of faults very small, less overlap
  - Correlation of faults can cause big problems

Example using the model

- How much does it help to shorten detection time?
- Portion of real archive (www.archive.org)
  - Monthly snapshots of web pages
  - 1.5 million immutable files
  - 1795 200GB SATA drives, “JBOD”
  - Mean time to visible (disk) failure: 20 hours
  - Almost 3 years of monthly file checksums
  - Mean time to latent fault 1531 hours

Scenario: audited replicated archive



Current and future work

- Using further modeling to choose
  - Auditing rates & patterns
  - Encoding and replication techniques
- Gather more failure data & introduce cost models
- Fire drill design
- Techniques for evolving
  - Metadata
  - Access controls
- Experiments with disk spin up/down reliability
- Building a low-power, high-density repository
  - For office/warehouse/home/trailer, not data center

Dynamic long-term architecture

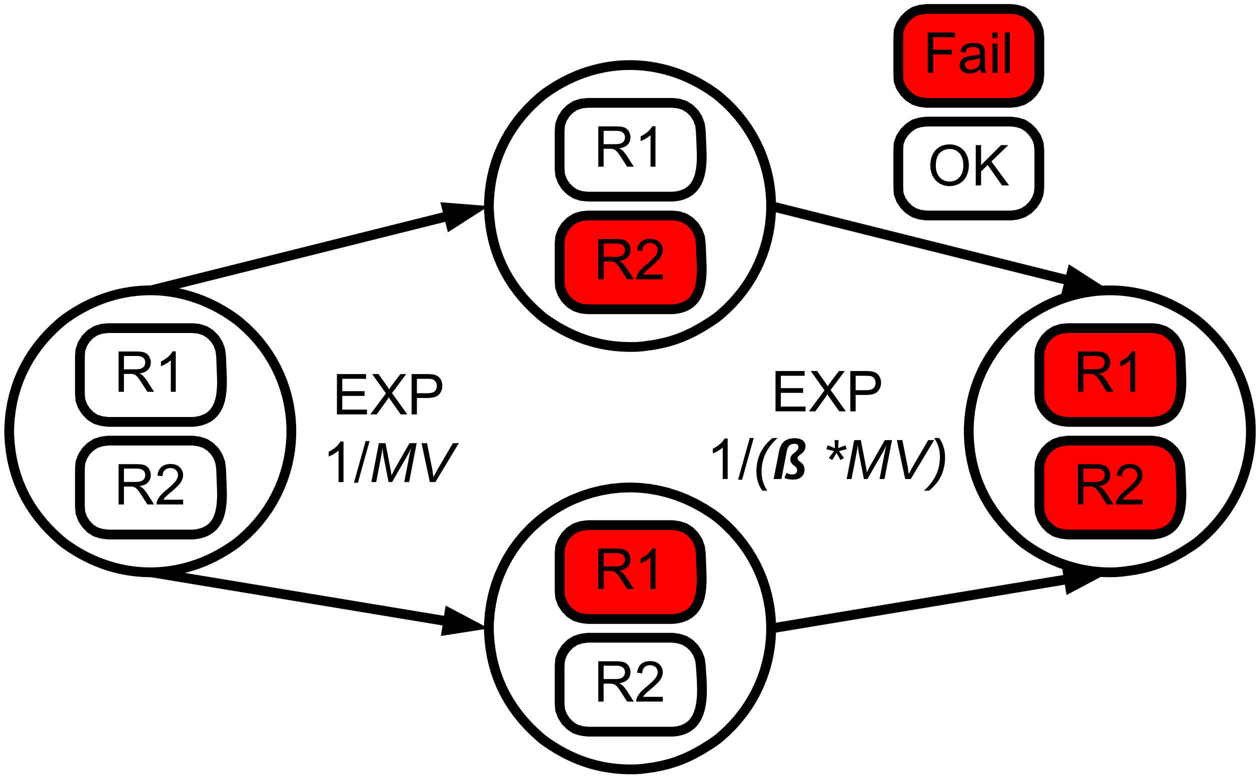
- Independent replicas
  - Geographic, administrative, platform
  - Gains from extra replication offset by correlations
- Inexpensive audit of content
  - Fix latent faults at all levels before they accrue
  - *Content must be accessible to do this cheaply!!*
  - Backup to high-latency off-line media is not a solution
  - Includes “repairing” endangered content/metadata
- Allow for on-going evolution of system
  - Components will always heterogeneous and changing
- Keeping data static requires a dynamic system!

Backup Slides

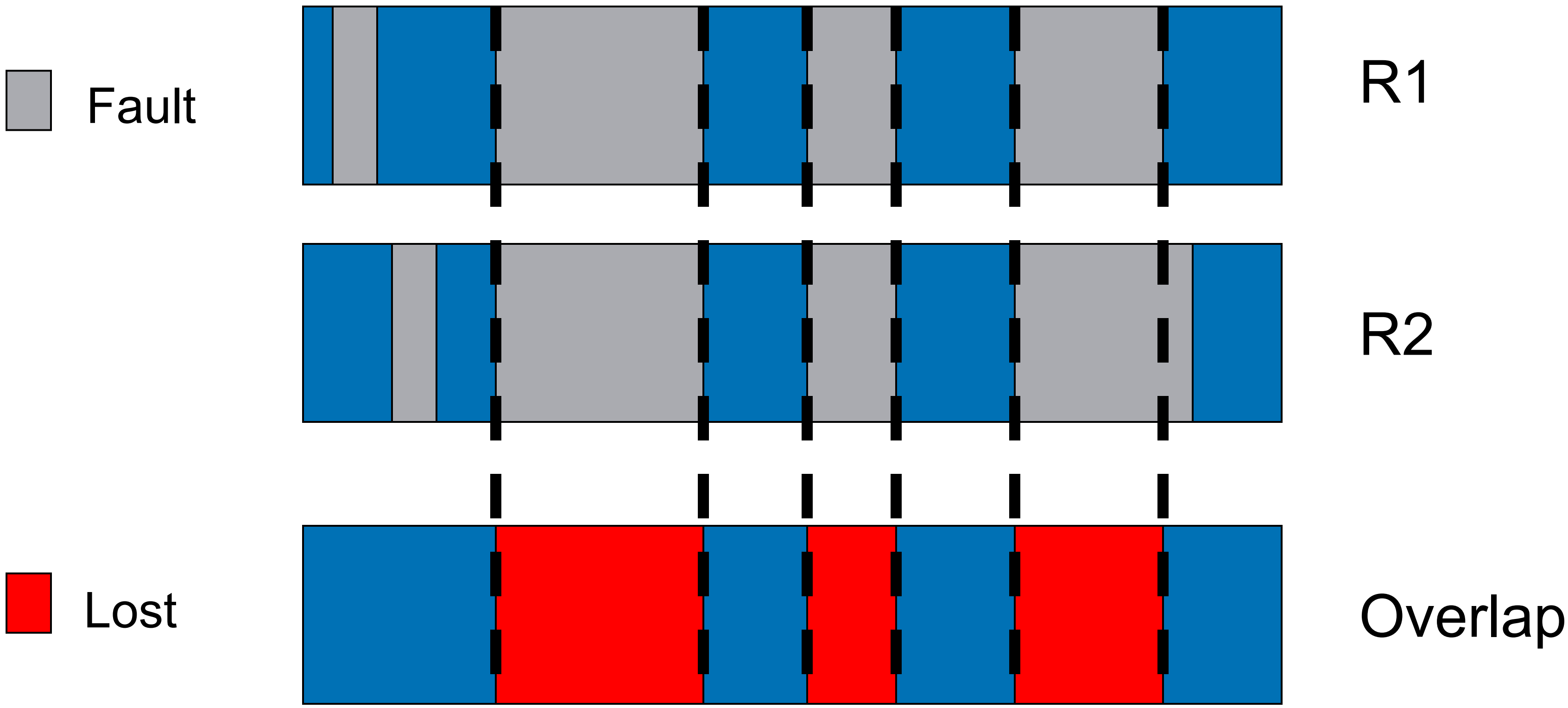
Commodity storage managed differently – up and down the stack

User APIs	Flexible presentation strategies over time Efficient ingestion and data exit strategies
On-going, automatic mgmt. processes	Metadata restructuring for continued usability, content repurposing Validation of access controls/roles Content migration to new formats/infrastructure End-to-end automatic audit and repair of visible and latent faults •Latent faults are a big threat in long-term content •Many sources: human error, attack, bit rot…
System	Replication across geographies, administrative boundaries •Replication essential for long-term reliability, low-cost audit Commodity file system, avoid dependence on external components
Storage containers	Low-power, high-density packaging •Mostly spun down
Storage	Use low-end, commodity, online storage •Currently disks/arrays

Temporally correlated faults



Spatial correlation



- Multiplicative fudge factor to express spatial correlation

Economic faults

- Budgets stretched just to ingest data
- Ongoing costs
  - Power
  - Cooling
  - Bandwidth
  - System administration
  - Equipment renewal
  - Domain registration
  - Space (rent)
- Lack of tools to predict these costs ahead of time
  - Harder to plan for longer lifetime
- It's the price/bit/year that matters

Attack

- We tend to worry about short-term intense attack
- Traditional repositories subject to long-term attack
  - Online repositories will be too
- Content destruction, censorship, modification, theft
  - Illegal or legal
  - External or internal
- Successful attacks may go unnoticed
  - Another example of a latent fault

Media/hardware obsolescence

- Media & hardware components become obsolete
  - Can’t communicate with other system components
  - Irreplaceable (or too expensive)
- Particularly acute for removable media
  - Readable but no suitable reader device

Human error

- Humans increasingly the cause of system failures
- Many ways for people to make mistakes
  - Accidentally remove/overwrite data
  - Accidentally mark data with incorrect permissions
  - Lose tapes in transit
  - Install bad device drivers
  - Etc.
- During archival lifetimes, assume this will occur
- Damage may go undetected

Component faults

- Take end-to-end view of storage system
  - Any component may fail
  - Hardware, software, firmware, network, ingestion, etc.
- With long-term view, add things like
  - 3rd-party license servers
  - Certificate authorities
  - URLs
  - Name services

Organizational faults

- Long-term view must include the organization
- Organizational structures die/merge/change
- Digital assets often invisible in reorgs/transfers
- Data vulnerable to single organizations/services

Software/format obsolescence

- Data still physically accessible/readable
- Cannot be interpreted
  - “RAW” formats of digital cameras
  - Early word processor formats
  - Compression/encryption formats
- Proprietary formats particularly vulnerable

Loss of context

- Information about the data
  - Layout
  - Inter-relationships between objects
  - Location
  - Provenance
  - Access restrictions
  - Necessary processes, algorithms, software
  - Database indices
- Encrypted data particularly vulnerable
  - Secrets get lost, leak or get broken