From What Are We Preserving Digital Information?

David S. H. Rosenthal

LOCKSS Program
Stanford University Libraries
http://www.lockss.org/
http://blog.dshr.org/

Fundamental Questions

- What are we to preserve?
- From what are we to preserve it?
- What's the budget?

Why LOCKSS?

- Fit the Organization to the Software?
  - Normally a recipe for disaster
- Fit the Software to the Organization!
  - Work for the Web the way libraries worked for paper
- Transparency
  - Don't make libraries re-train their users
  - Don't break the links!
- Lots of Copies
  - What can be done with them to reduce cost of ownership?

What Can Possibly Go Wrong?

- Media Failure
- Hardware Failure
- Software Failure
- Network Failure
- Media & Hardware Obsolescence
- Software Obsolescence
- Operator Error
- Natural Disaster
- External Attack
- Internal Attack
- Economic Failure
- Organizational Failure

Principles of Fault-Tolerance

- Byzantine Fault Tolerance (BFT)
  - 3f+1 replicas survive f simultaneous faults
  - Replicas don't trust each other, they all vote on any action
- Failures are correlated
  - Hard to predict how often you get f simultaneous faults
- Reduce correlation: loose coupling
  - BFT: all replicas vote
  - LOCKSS: a sample of the replicas vote

What Can Possibly Go Wrong?

- Media Failure
- Hardware Failure
- Software Failure
- Network Failure
- Media & Hardware Obsolescence
- Software Obsolescence
- Operator Error
- Natural Disaster
- External Attack
- Internal Attack
- Economic Failure
- Organizational Failure

Media & Hardware Failures

- Idea: failed box replaced by new empty box
  - Content automatically recovered from the network
  - No need to back up, no need to recover from backup
- Disks got bigger faster than net got faster
  - Recovery works, but now too slowly
  - Working to speed up, re-configure box storage
- Disks got bigger faster than more reliable
  - RAID rebuild now likely to lose some data
  - Recover most via RAID, rest via LOCKSS works well

Software Failures

- Ideal: boxes run independently written code
  - Too expensive for non-life-critical applications
  - Even then not completely independent
- LOCKSS: boxes run same code differently
  - Box behavior heavily randomized
- Possible: stagger software updates
  - Releases: 1/3 N, 1/3 N-1, 1/3 N-2
  - Downside: 6 week cycle, oldest code <= 24 weeks old

Network Failures

- Networks fail on timescales short and long
  - Boxes collect independently, then poll to reach agreement
  - Polls happen slowly, and are in any case sampled
- LOCKSS net infrastructure: props server
  - Props treated as an AU to be preserved
  - Delay in access not critical
- LOCKSS box avoids using network services
  - Doesn't use DNS to locate other boxes
  - Doesn't use PKI to distribute certificates
  - Too fragile to depend on in the long term

Media & Hardware Obsolescence

- Failures just early obsolescence
  - Survive them the same way
  - Most obsolescence is from interface evolution
- Box-to-box interface is Ethernet & TCP/IP
  - RJ11 very stable because it is in the walls
  - TCP/IP very stable – no flag day on the Internet
  - Note timescale for IPv6 introduction

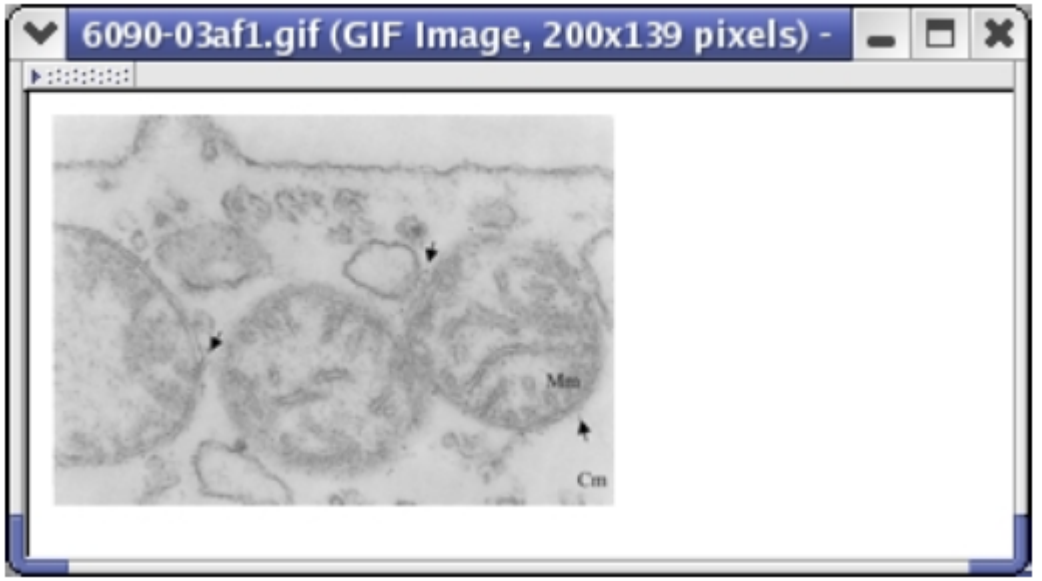
Software Obsolescence

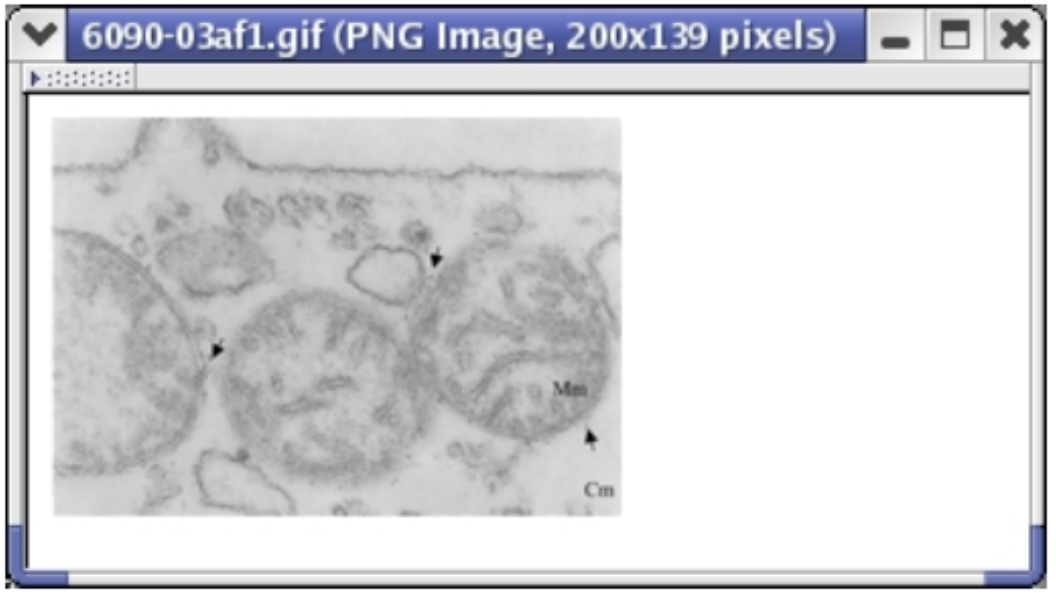
- Effect of software obsolescence
  - Format obsolescence
  - LOCKSS attitude to format obsolescence controversial
- Web formats are not going obsolete
  - Hard to find formats that have gone obsolete since 1995
- Web makes formats a publishing medium
  - Format obsolescence = remove format from browser
  - Doing so makes browser *worse*
  - No economic driver for format removal

LOCKSS & Format Obsolescence

- LOCKSS preserves (only) the original bits
  - Migration enthusiasts keep the migrated version
  - But are never confident enough to discard the original
- LOCKSS uses HTTP content negotiation
  - Browser tells server which formats acceptable
  - If original format not acceptable, trigger migration
- Daemon finds converter to acceptable format
  - Serves migrated file with original name, new mime-type
  - Discards (maybe caches) the migrated file
  - Demo-ed GIF to PNG migration in 2004

Format Migration Demo





What Can Possibly Go Wrong?

- Media Failure
- Hardware Failure
- Software Failure
- Network Failure
- Media & Hardware Obsolescence
- Software Obsolescence
- Operator Error
- Natural Disaster
- External Attack
- Internal Attack
- Economic Failure
- Organizational Failure

Operator Error

- Eliminate opportunities for error
  - E.g. original OpenBSD live CD
  - Good: cheaper, operator error less likely
  - Bad: easier to ignore box, not notice problems
- Limit the scope of error
  - Each box under separate administration
  - No-one with admin access to multiple boxes
  - We violate this rule in early days of networks

Natural Disaster

- Natural disasters restricted geographically
  - We hope ...
- LOCKSS boxes natural geographic diversity

- Props server backup in the Amazon cloud
  - We use it frequently to cove software upgrades, etc.
  - Should probably have a backup in another cloud too

External Attack

- LOCKSS defensive design
  - Protocol resists attacks to modify content or deny service
  - Box-to-box communication encrypted via SSL
  - No other applications on the box
  - Only non-firewalled port should be the protocol port
- OpenBSD live CD was designed for defense
  - Software either on read-only media
  - Or signatures checked by software on read-only media
  - No place to put a Trojan that would survive reboot
- O/S diversity would be good
  - Supporting OpenBSD, Linux, Solaris got too costly

Internal Attack

- By the admin of a box
  - Provided 1 admin 1 box rule followed
  - Can only use box to attack via protocol - hard
- By the admin of the props server
  - Worst-case props settings can only deny service
- By the LOCKSS developers
  - Inherent vulnerability of single code base
  - Staggered software updates would help
  - At cost of delay in fixing bugs

Economic Failure

- By far the biggest risk to preserved content
  - E.g. “Blue Ribbon Task Force” report
- LOCKSS started with grant funding
  - Small grant from NSF for prototype
  - Mellon, NSF, Sun funded production code
- LOCKSS Alliance sustainable since 2007
  - Thanks to 2005-7 transition assistance from Mellon

Organizational Failure

- LOCKSS is not core mission for Stanford
  - If we cost money or cause problems, we'd be out
- Another home would be needed for:
  - Software maintenance
  - GLN content processing
  - Props servers
- Tricky but not impossible
  - Not a great deal of work

Overall System Reliability

- Petabyte for a century example
  - With 50% chance of every bit surviving undamaged
  - Bit half-life 60 million times age of universe
  - Or, 10 million times as reliable as Amazon's S3
- From a 2006 talk at SDSC
  - To a 2010 paper in *Communications of ACM*
- Fundamental observations
  - Required reliability too high to measure directly
  - Simulations based on component reliability unrealistic
  - We're never going to be sure that our systems work

Cloud Storage

- I'm working on cloud storage for LOCKSS
  - Funded by Library of Congress
- Some cloud storage configurations working
  - Details – see my NDSA talk
- Basic question “is it cheaper?”
  - Local storage has purchase + running costs
  - Cloud storage has only running costs
  - Need apples to apples comparison
  - Must compare costs *through time*

Discounted Cash Flow

- Costs now vs. future costs?
  - Assume an interest rate
  - Invest now so that principal + interest = future cost
  - Amount invested now is net present value of future cost
- What interest rate to use?
  - If certain about future, use Treasury bond rates
  - The less certain, the more *risk premium* added to rate
- This is the standard technique investors use
  - What could possibly go wrong?

Does DCF Work In Practice?

First, there is statistically significant evidence of short-termism in the pricing of companies' equities. This is true across all industrial sectors. Moreover, there is evidence of short-termism having increased over the recent past. Myopia is mounting. Second, estimates of short-termism are economically as well as statistically significant. Empirical evidence points to excess discounting of between 5% and 10% per year.

- A. Haldane & R. Davies *The Short Long*

Does DCF Work In Theory?

What this analysis makes clear, however, is that the long term behavior of valuations depends extremely sensitively on the interest rate model. The fact that the present value of actions that affect the far future can shift from a few percent to infinity when we move from a constant interest rate to a geometric random walk calls seriously into question many well regarded analyses of the economic consequences of global warming. ... no fixed discount rate is really adequate – as our analysis makes abundantly clear, the proper discounting function is not an exponential.

– D. Farmer & J. Geankoplos *Hyperbolic Discounting is Rational: Valuing the Far Future With Uncertain Discount Rates*

Economics of Long-Term Storage

- A single interest rate is just wrong
  - So is a single rate of $/byte decrease
- Need to follow possible paths through future
  - Varying interest rates
  - Varying technology costs and service lives
- Need a Monte Carlo simulation
  - Follow many paths
  - Chose interest rates, costs, service lives from distributions
  - Average over the paths to get:
    - Most probable outcomes, range of outcomes

Components of the Model

- Yield Curves
  - Relate duration of loan to interest rate at given time
- Loans
  - Principal, interest rate, duration, $ outstanding
- Assets
  - Principal, interest rate, duration
- Technologies
  - Purchase cost, running cost

Technology

- Each technology has:
  - *Purchase cost*: e.g. for disk follows Kryder's Law
  - *Running cos*t: e.g. for cloud, rental, bandwidth, compute
  - *Move-in cost*: e.g. for cloud upload bandwidth cost
  - *Move-out cost*: migration cost = move-out + move-in
  - *Service life*: e.g. for disk 4 or 5 years
- If model chooses to deploy a technology:
  - *Purchase loan*: pays for purchase and migration cost
  - Duration of loan = service life, interest from yield curve

Operations of the Model

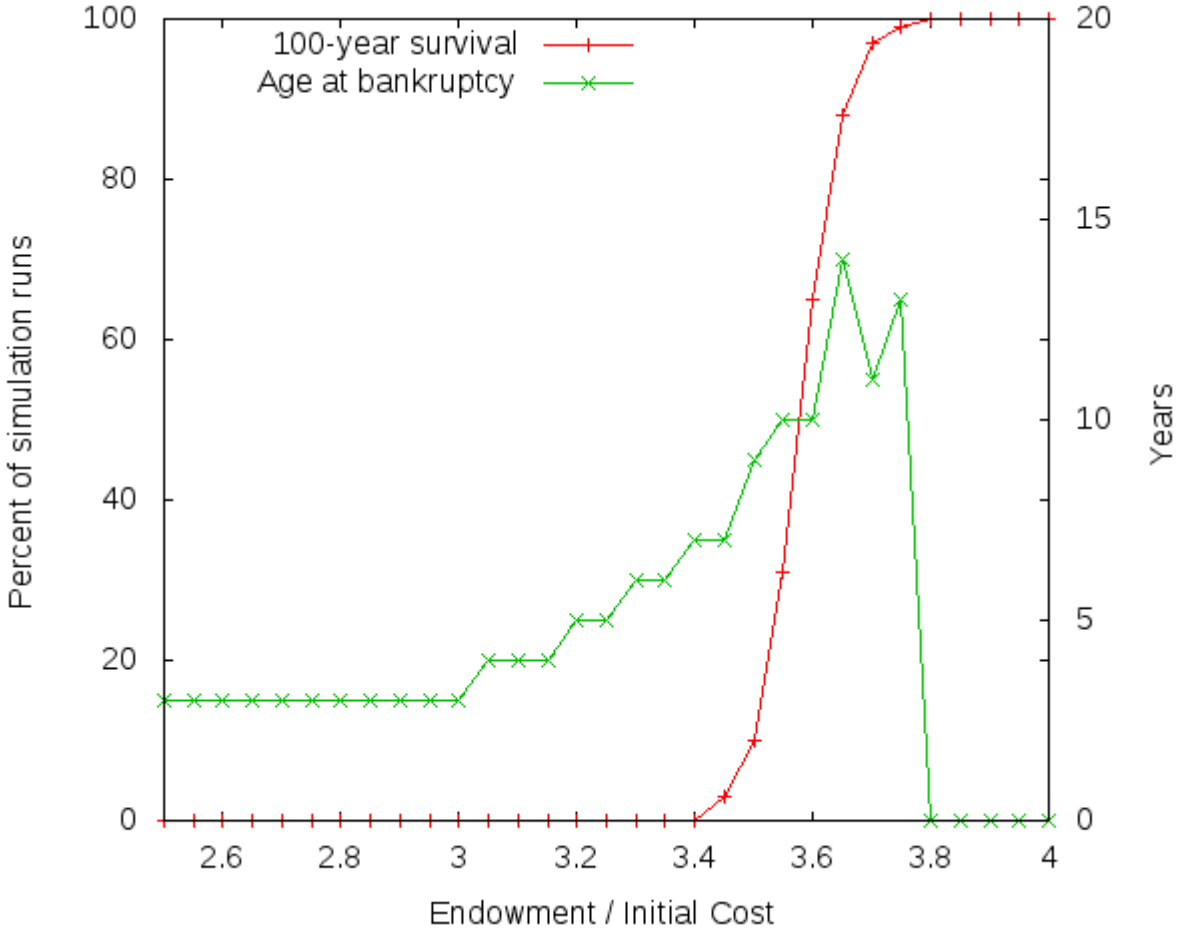
- Set yield curve for the year
  - Start at a random year in the past
  - Run time forward then backward then forward then ...
- Create some new technologies
  - If better, replace existing technologies in available set
- For each deployed technology:
  - Run the hardware upgrade process
  - Pay running cost and purchase loan cost
- Pay any outstanding purchase loans
  - From prematurely retired technologies

Hardware Upgrade Process

- If T.old at end of service life
  - Set period P to planning horizon, cost C(T.old) to infinity
- Else
  - Set P to min(remaining service life, planning horizon)
  - Compute cost C(T.old) over P
- For each available technology T.new
  - Compute cost C(T.new) over P
- If smallest C(T.new) less than C(T.old)
  - Replace T.old by T.new in set of deployed technologies

Example of Using the Model

- Kryder's Law
  - + running cost
- Interest rates
  - From 1990-2010
- Endow data
  - % survive?
  - How long to $0?



Other Uses of the Model

- Effects of short-termism on storage cost?
  - How long a planning horizon should we use?
- Effect on endowment of pauses in $/byte?
- Low running costs vs. low purchase costs?
  - E.g. flash vs. disk?
- Local disk vs cloud storage?
  - Purchase + run costs vs. run cost only?

Obvious Improvements

- Plug-able alternate models for:
  - Interest rates
  - Technology evolution
- Plug-able alternate policies for:
  - Investing the endowment
  - Upgrading the hardware
- Implement as a web site
  - To allow “what-if” scenario planning