Distributed Digital Preservation with LOCKSS

Nicholas Taylor (@nullhandle)
Program Manager, LOCKSS and Web Archiving
Stanford Libraries

Massive Storage Systems and Technology
14 May 2018
overview

• LOCKSS background
• preservation principles
• distributed preservation
• what’s next for LOCKSS?
(digital) libraries
lots of copies (were already) keeping stuff safe

- print journal holdings incidentally resilient:
  - distributed
  - decentralized
  - irrevocable
  - tamper-evident
  - publisher-independent
move to online publishing

• own → lease
• local custody → contingent access
• lots of distributed copies → fewer, centralized copies
• net effect:
  • endanger scholarly record
  • obviate library role
LOCKSS but for e-journals

- **p2p software** for e-journal preservation
- **restore preservation features** of print journal holdings for digital
- **re-empower libraries**, individually + communally
- **improve durability of digital scholarly record**
LOCKSS for more than e-journals

- set out to build e-journal preservation system
- ended up building generic digital preservation core
- growing number of communities use to preserve other digital materials
community + digital preservation

- communities complement LOCKSS:
  - resilience against organizational failure
  - native heterogeneity

- preservation is an active community effort

- lots of communities keep stuff safe
Preservation Principles
lots of copies

• intuitive best practice
• LOCKSS typically operates w/ 4+ copies
• enlist copies to attest to expected integrity value
• lots of copies enables:
  • majority votes w/ minority of participating copies
  • higher-confidence attestations via landslide agreement
routine audit + repair

• ensuring long-term bit integrity
  • must read data to know it’s good
  • easier to repair data sooner
• network nodes conduct polls to validate integrity of distributed copies
• more nodes = more security
  • more nodes can be down
  • more copies can be corrupted
  • ...and polls will still conclude
• nonces force re-hashing
• peers are mutually-distrusting
fail slowly

- fast-operating, tightly-coupled systems fail quickly
- LOCKSS is conservative + sophisticated about repairs
- polls run slow to enable detection + mitigation of cause of damage
threat model

• familiar threats:
  • media + hardware obsolescence
  • software obsolescence
  • natural disaster

• more typical threats:
  • economic failure
  • organizational failure
  • operator error

• security threats:
  • internal attack
  • external attack
distributed + decentralized

- no monopoly on copy-making
- more copies doesn’t mitigate correlated risk
- independent, decorrelated copies
- minimize central points of failure or vulnerability
no centralized fixity store

- fixity data subject to same threats as data whose integrity it assures
- fixity data is more vulnerable, in fact, since more valuable + more centralized
- LOCKSS uses fixity data in limited ways
local custody

- if preserving data is **core to mission**, LOCKSS helps maintain that competency + commitment in-house
- **unencumbered access** for use by designated community
- **conserving autonomy** + leverage w/ content + service providers
- **jurisdictional transparency** + control
Distributed Preservation

“Catho longtime [explored]” by Bill Collison under CC BY-NC 2.0
where does distributed preservation fit?

• may be integrated into own infrastructure (e.g., offsite replication)
• as a wholly hosted service:
  • for some, may be main preservation solution
  • for others, may supplement local preservation
use cases

• scholarly record
• government documents
• web archives
• collaborative collections
• any types of content valued in common by a community
distributed preservation providers

- **hosted services** w/ varied architectures, service tiers, levels of assurance, replication factors
- replication nodes include memory orgs + cloud
- none (including LOCKSS) require local preservation infrastructure
- LOCKSS provides opportunity for co-preservation
What’s Next?
re-architecture rationale

• de-silo + enable external integrations
• foster developer community
• capitalize on work of broader communities
• create space for system enhancements
• evolve w/ web + digital preservation ecosystem

“New York Reflection” by Reto Fetz under CC BY-NC-SA 2.0
anticipated outcomes

- functional parity + backward compatibility
- components providing value outside of end-to-end system
- better integration + data hand-offs w/ other apps
- increased use to preserve repository content
- increased use to preserve content managed by non-memory institutions
Questions