Making the Archive Real

Hakim Weatherspoon
University of California, Berkeley

ROC/OceanStore Retreat, Lake Tahoe. Tuesday, June 11, 2002

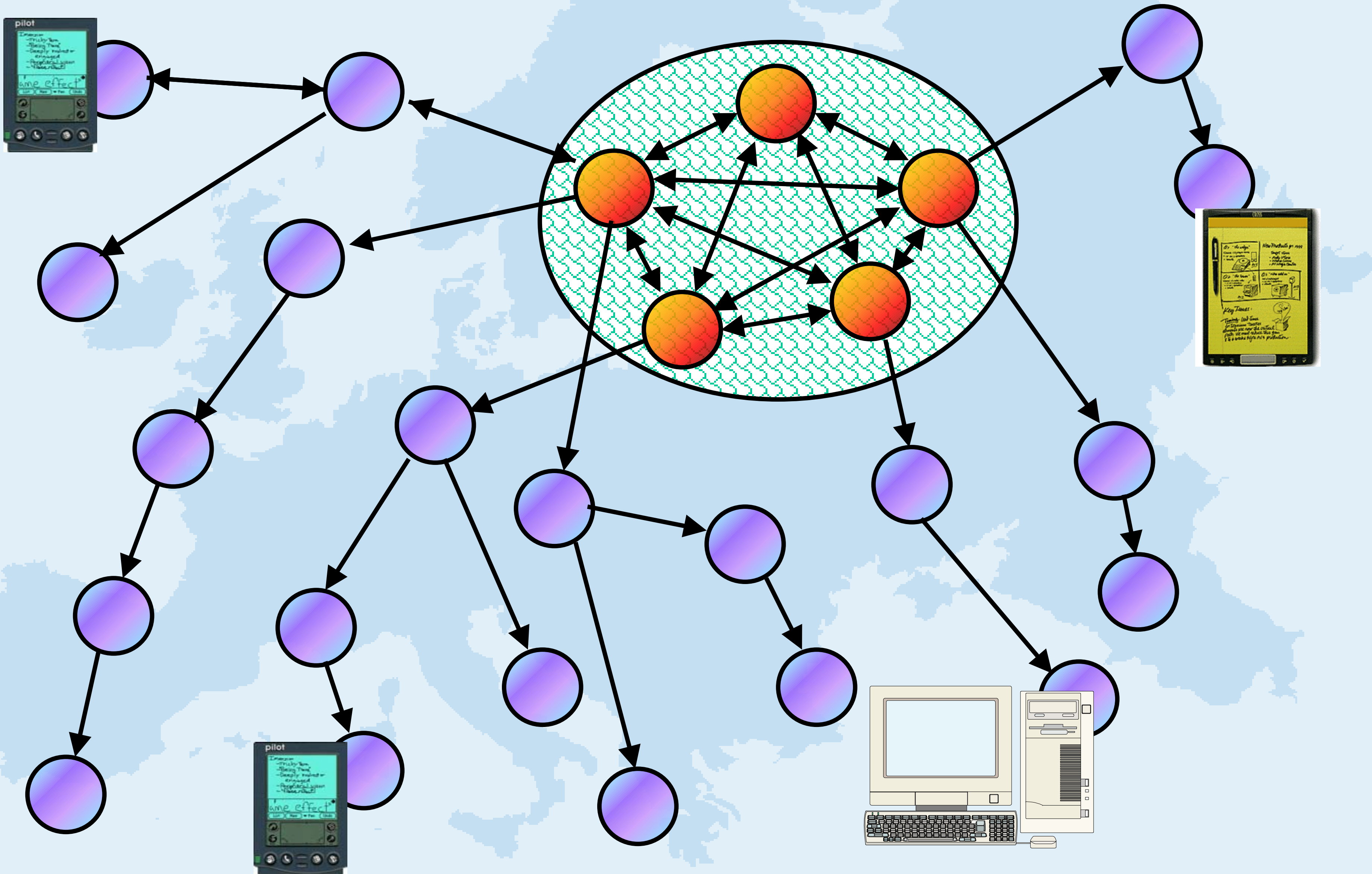
Archival Storage

- Every version of every object is archived.
  - *At least in principle.*
- How long does it take to…
  - *Write?*
  - *Read?*
- Can the Archive be run *inline*?
  - Or, is it best to batch?

Outline

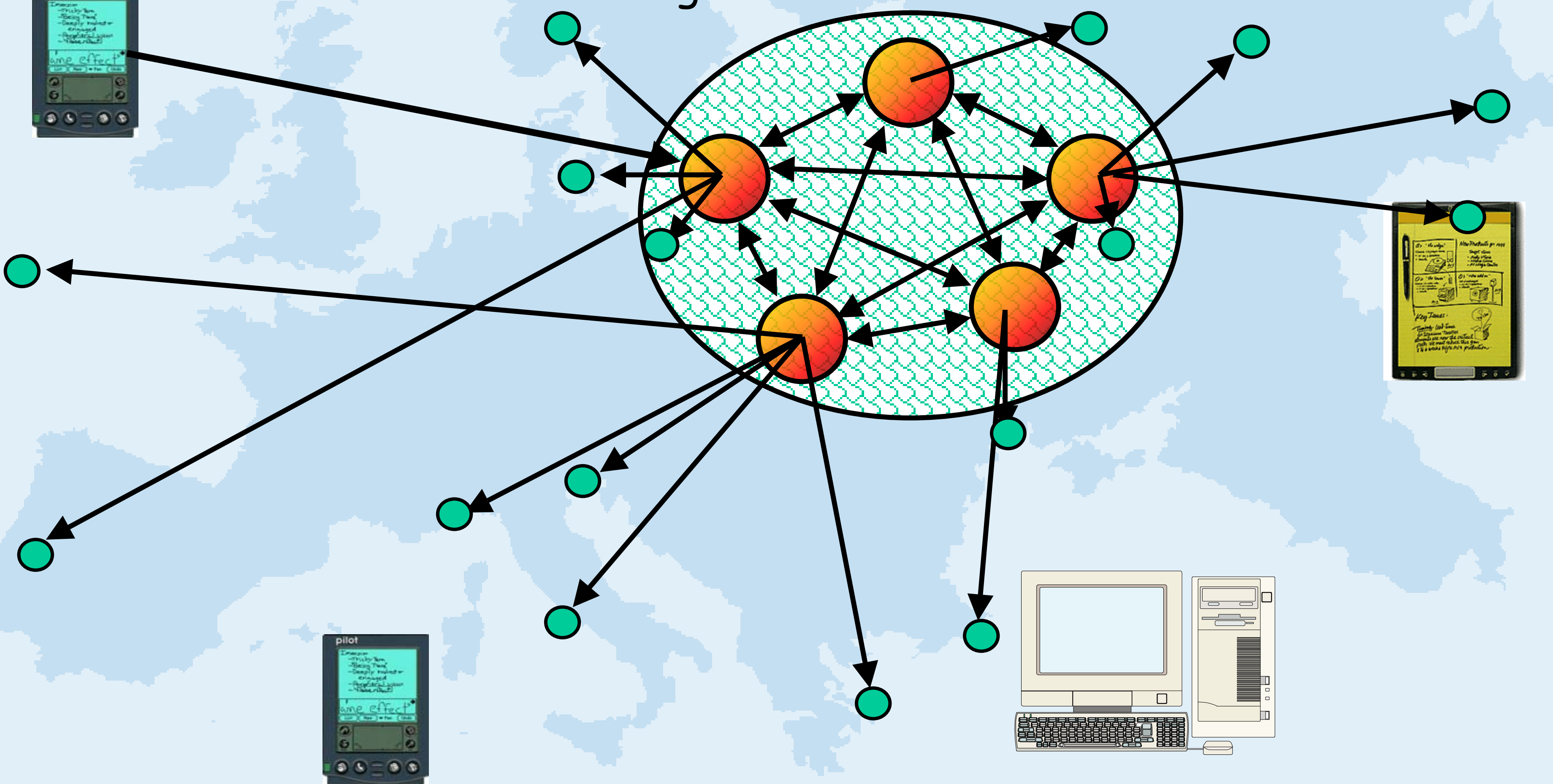
- Archival Process context.
- Archival Modes.
- Archival Performance.
- Future Directions

Path of an Update



Archival Dissemination Built into Update

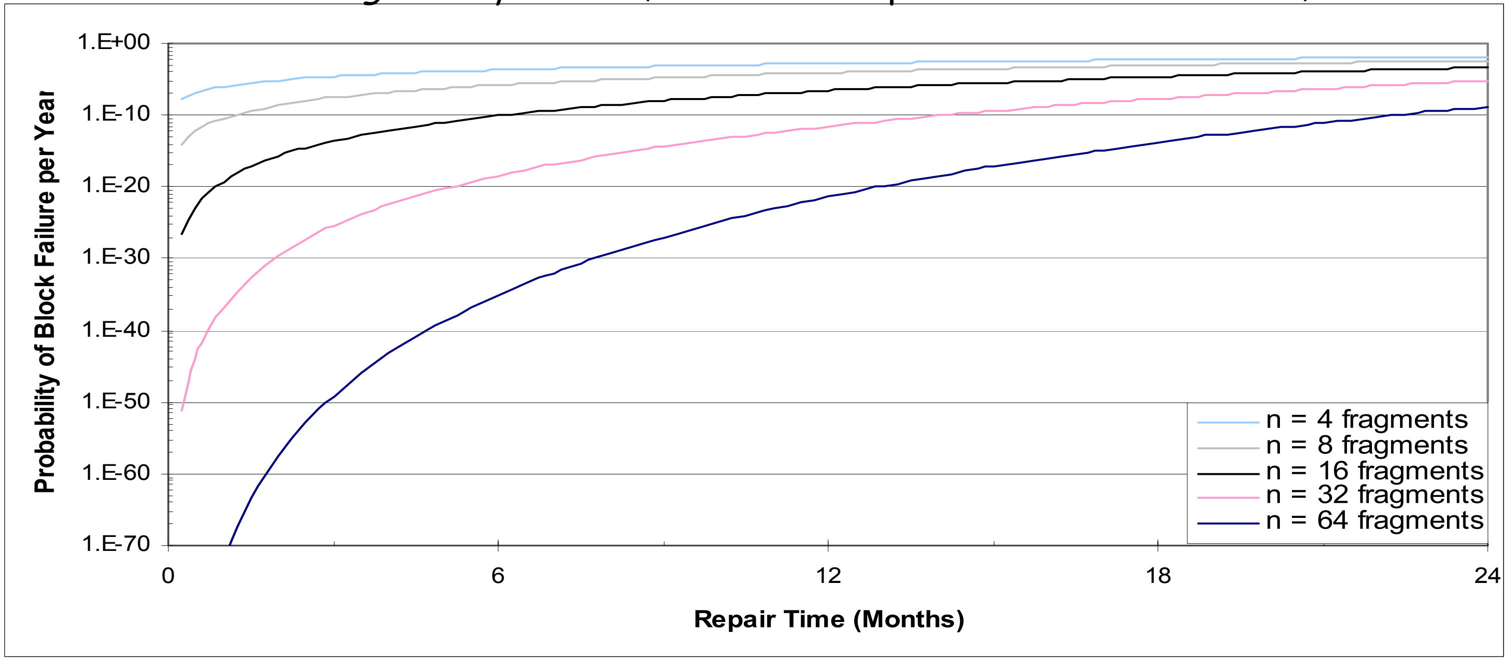
- *Erasure codes*
  - redundancy without overhead of strict replication
  - produce *n* fragments, where any *m* is sufficient to reconstruct data. $m < n$. rate $r = m/n$. Storage overhead is $1/r$.



Durability

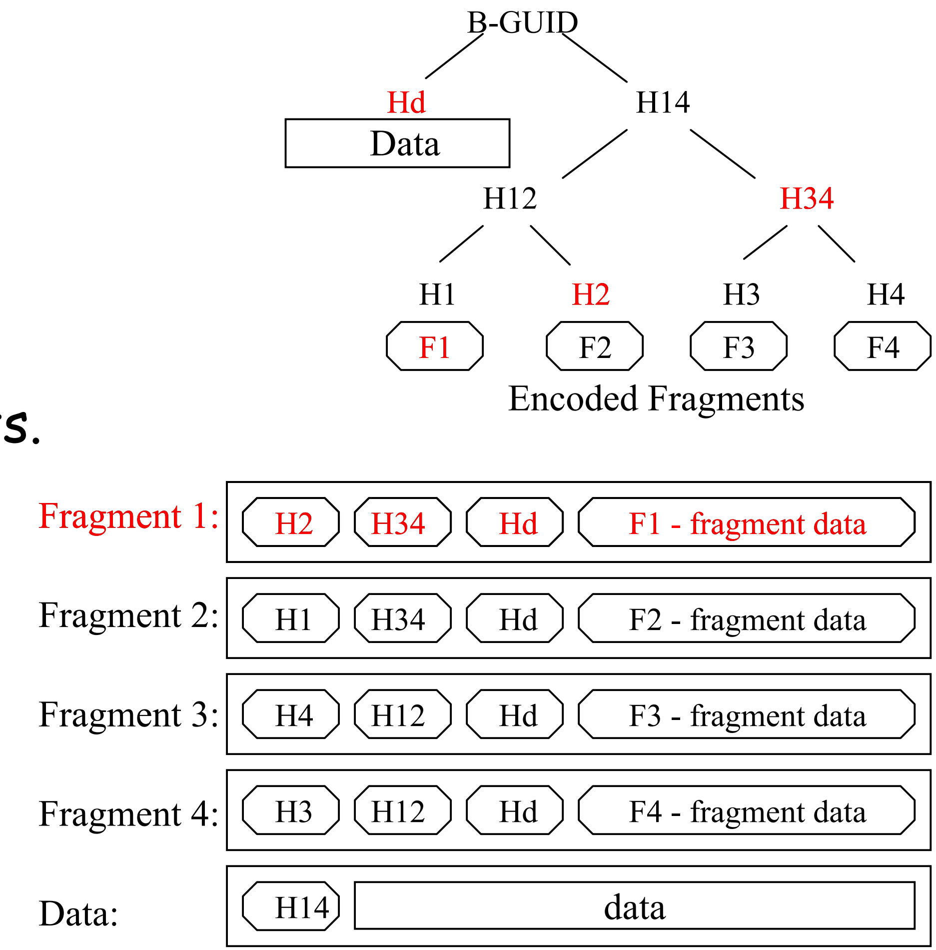
- Fraction of Blocks Lost Per Year (FBLPY)*
  - $r = \frac{1}{4}$, erasure-encoded block. (e.g. $m = 16$, $n = 64$)
  - Increasing number of fragments, increases durability of block
    - Same storage cost and repair time.
  - $n = 4$ fragment case is equivalent to replication on four servers.

* *Erasure Coding vs. Replication*, H. Weatherspoon and J. Kubiatowicz, In Proc. of IPTPS 2002.

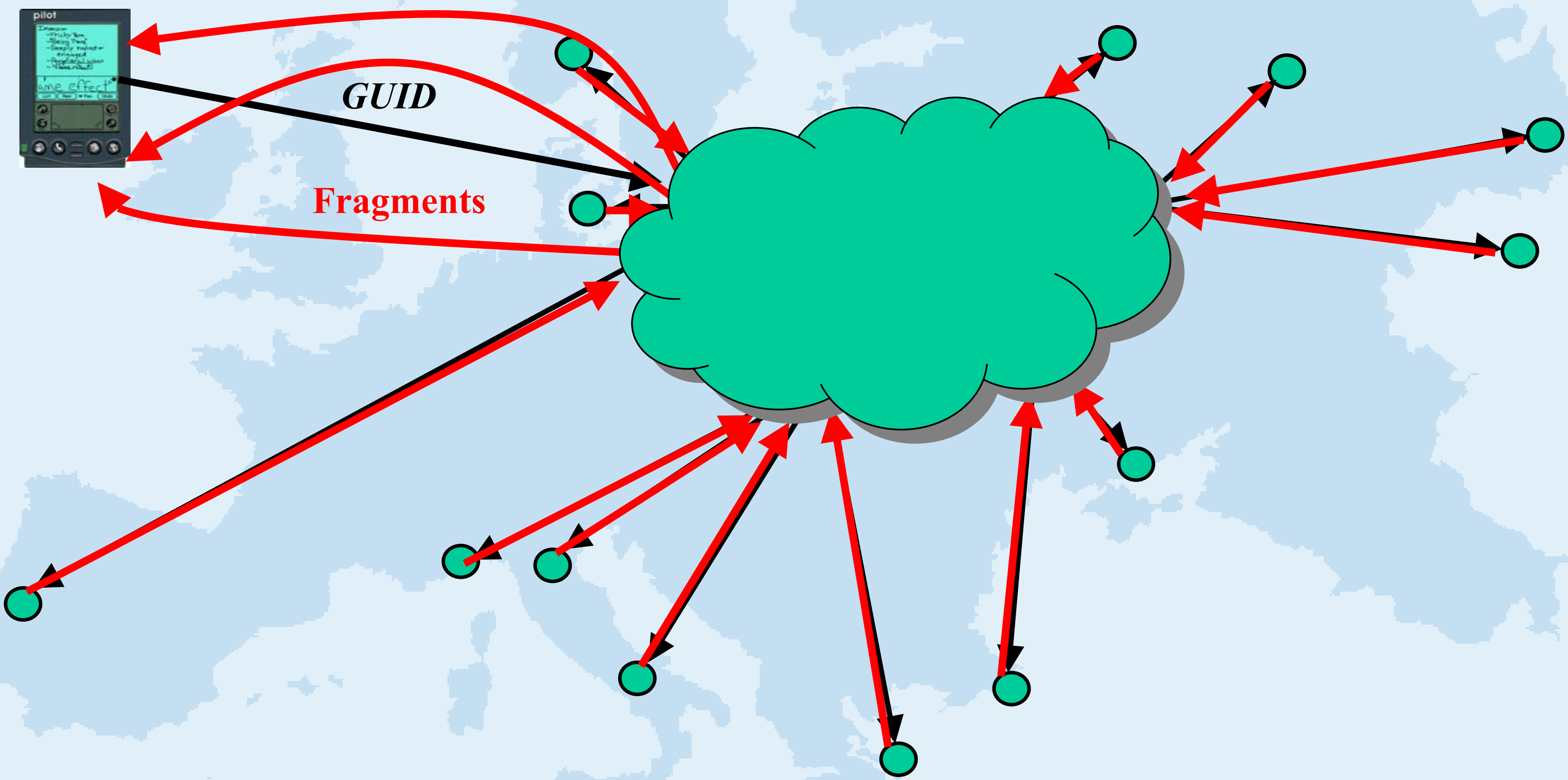


Naming and Verification Algorithm

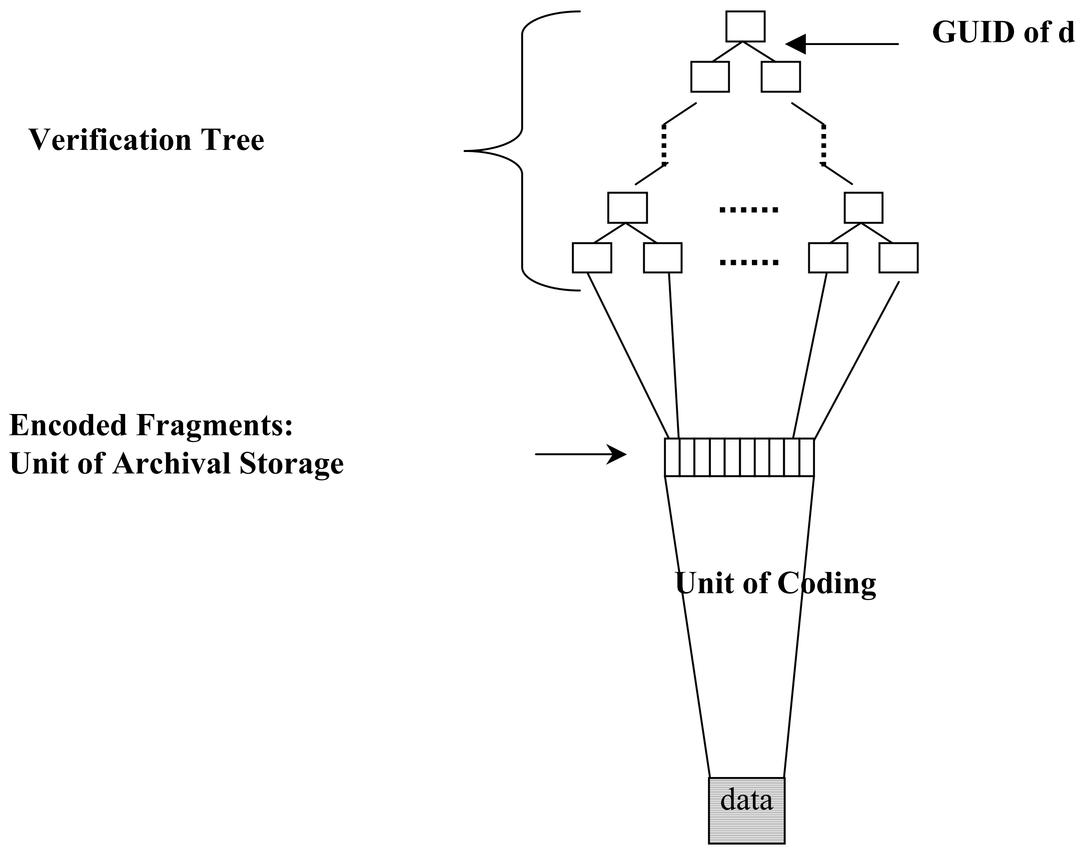
- Use cryptographically secure hash algorithm to detect corrupted fragments.
- Verification Tree:
  - *n* is the number of fragments.
  - store log(*n*) + 1 hashes with each fragment.
  - Total of *n*.(log(*n*) + 1) hashes.
- Top hash is a *block GUID* (*B-GUID*).
  - Fragments and blocks are self-verifying



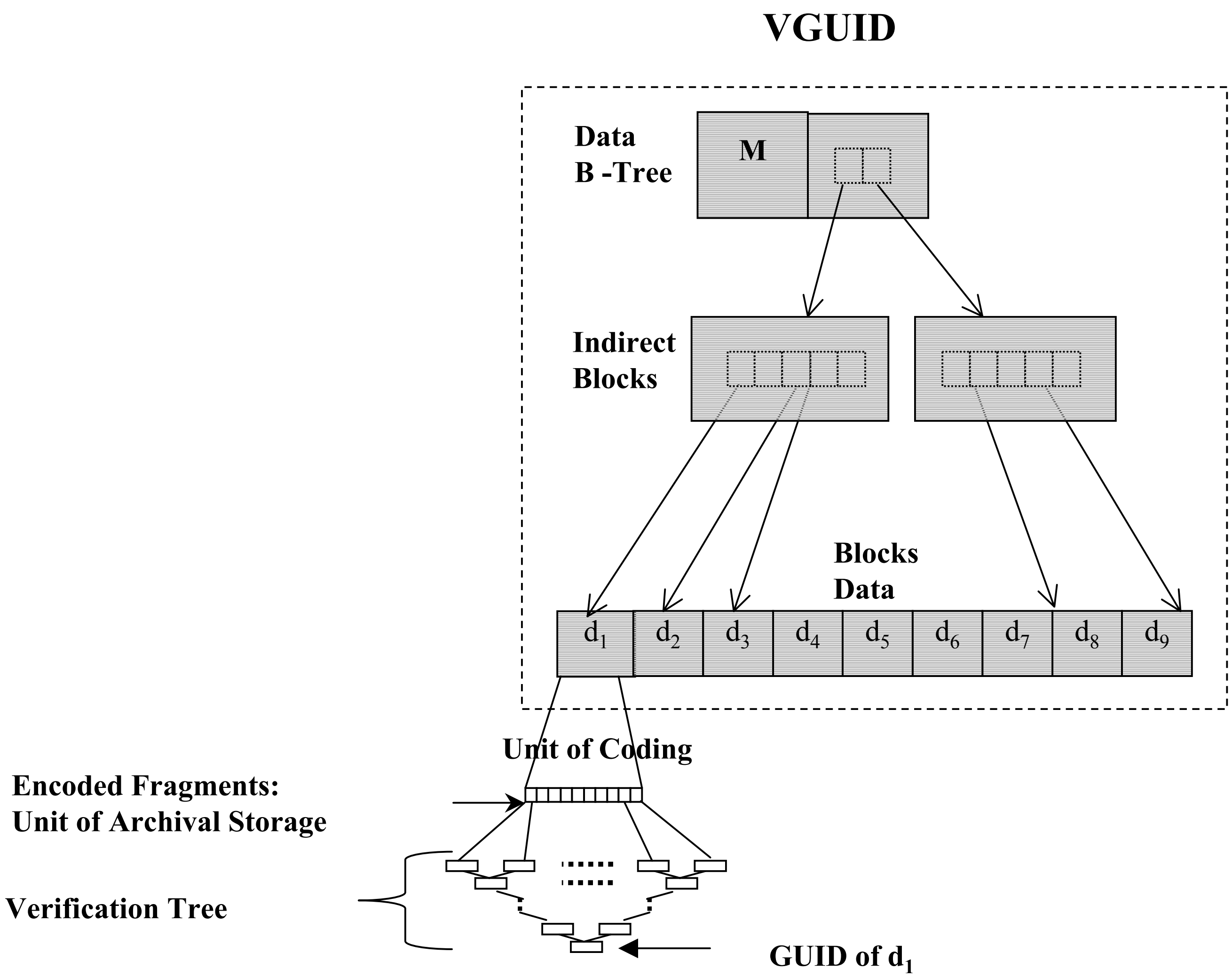
Enabling Technology



Complex Objects I

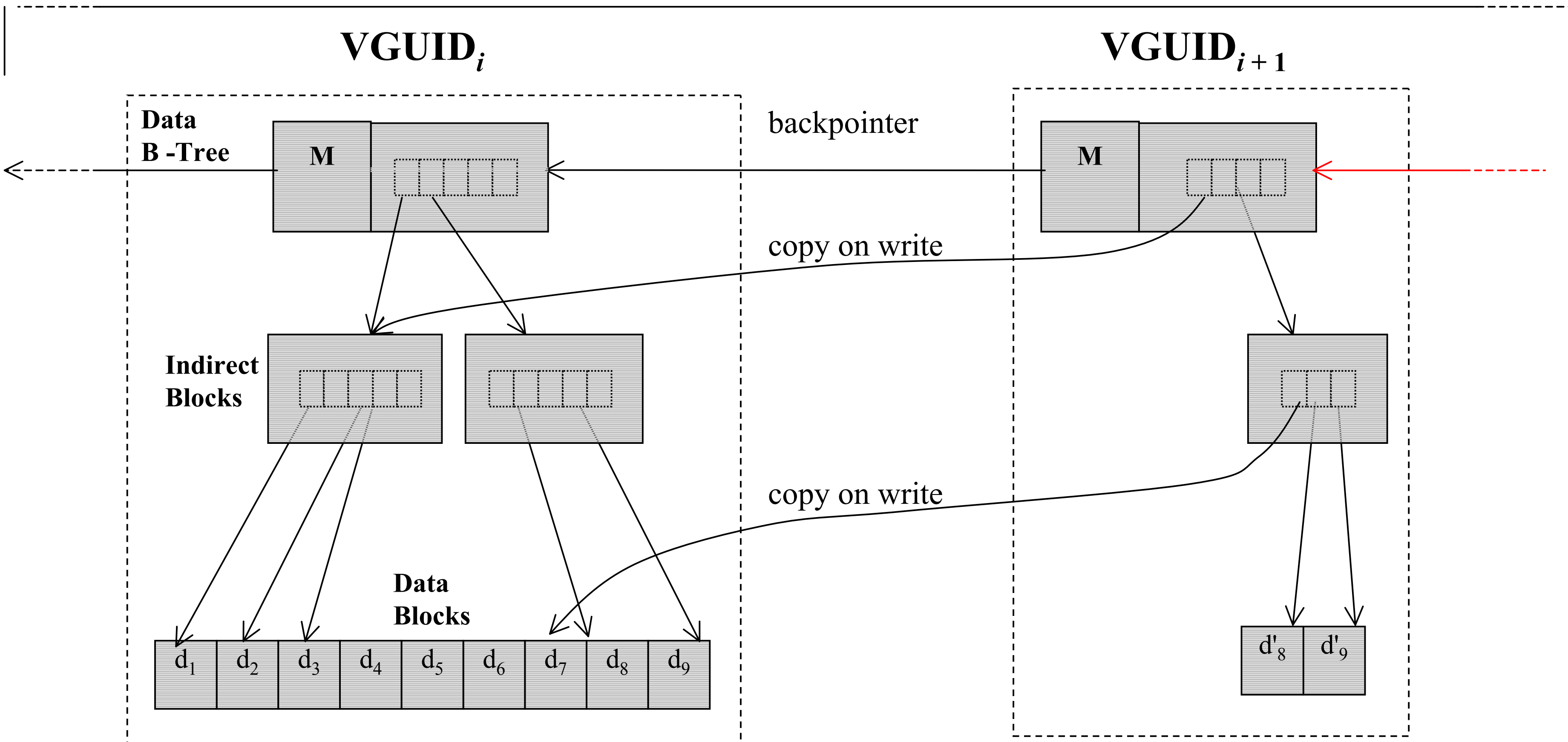


Complex Objects II



Complex Objects III

AGUID = hash{name+keys}

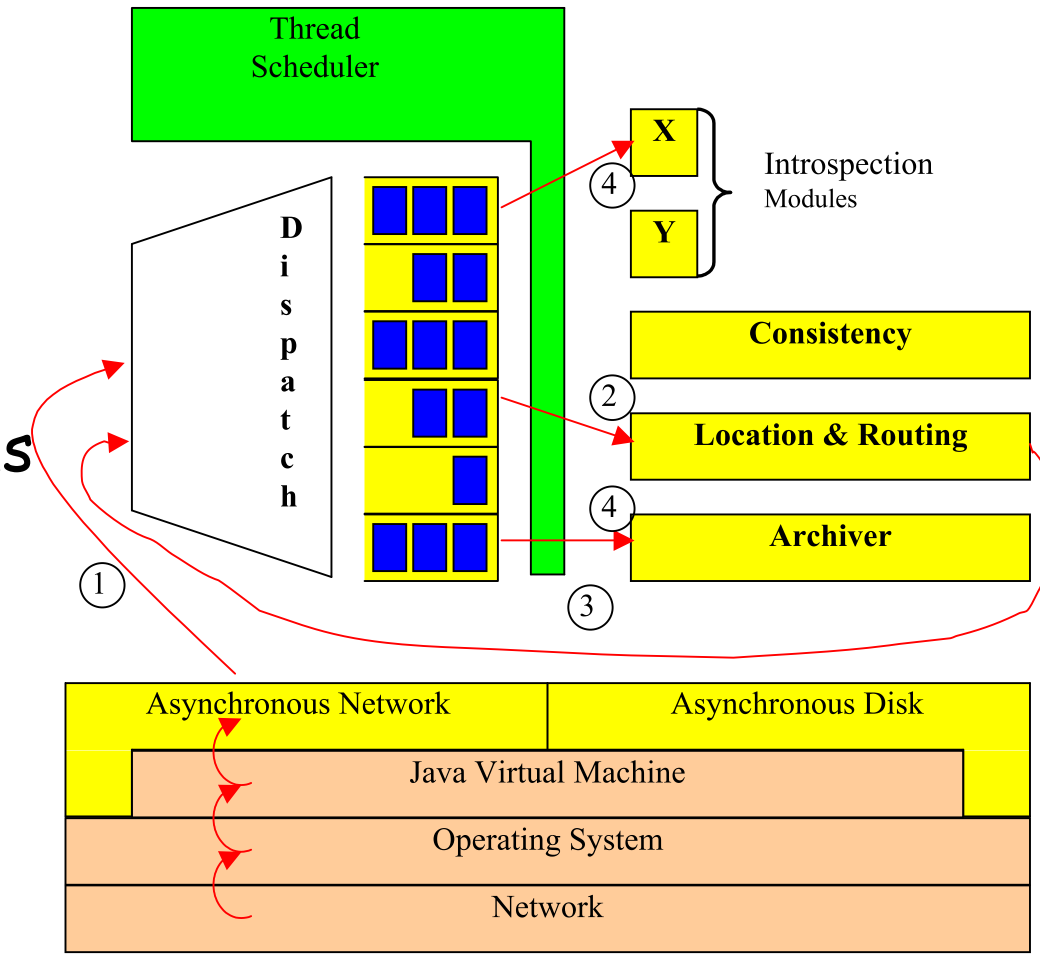


Mutable Data

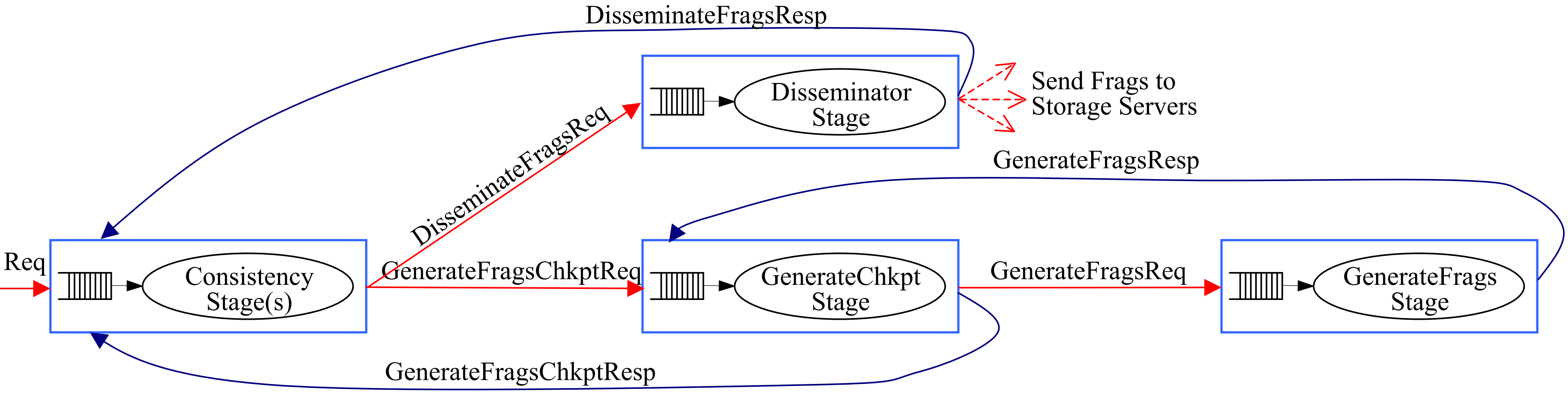
- Need mutable data for real system.
  - Entity in network.
  - A-GUID to V-GUID mapping.
  - Byzantine Commitment for Integrity
  - Verifies client privileges.
  - Creates a serial order.
  - Atomically applies update.
- Versioning system
  - Each version is inherently read-only.

Archiver I

- Archiver Server Architecture
  - Requests to archive objects recv'd thru network layers.
  - Consistency mechanisms decides to archive obj.



Archiver Control Flow



Archival Modes

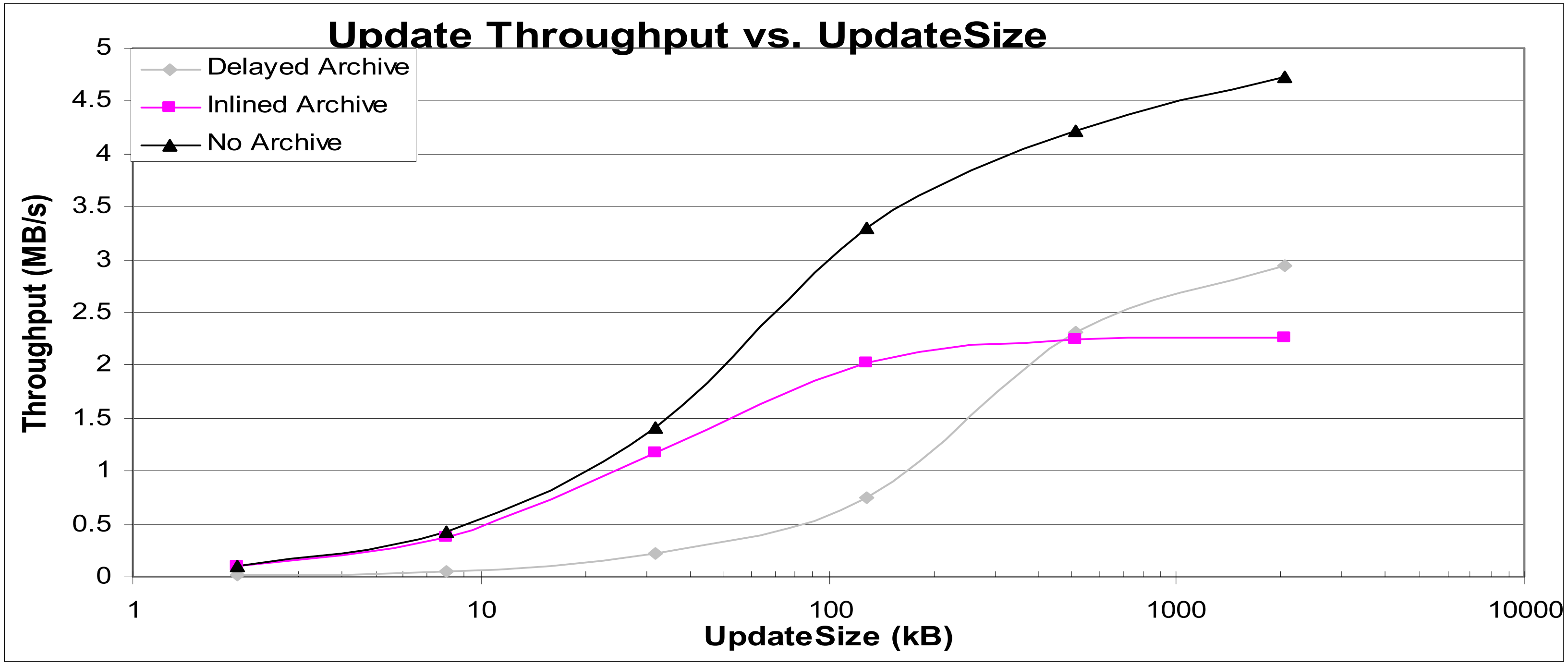
Mode	synch?	latency	durability
No Archive	synch	minimum	none
Inlined Archive	synch	maximum	maximum
Delayed Archive	asynch	minimum	medium

Performance

- Performance of the Archival Layer
  - Performance of an OceanStore server in archiving a objects.
  - analyze operations of archiving data (this includes signing updates in a BFT protocol).
  - $m = 16$, $n = 32$
- Experiment Environment
  - OceanStore servers were analyzed on a 42-node cluster.
  - Each machine in the cluster is a
    - IBM xSeries 330 1U rackmount PC with
    - two 1.0 GHz Pentium III CPUs
    - 1.5 GB ECC PC133 SDRAM
    - two 36 GB IBM UltraStar 36LZX hard drives.
    - The machines use a single Intel PRO/1000 XF gigabit Ethernet adaptor to connect to a Packet Engines
    - Linux 2.4.17 SMP kernel.

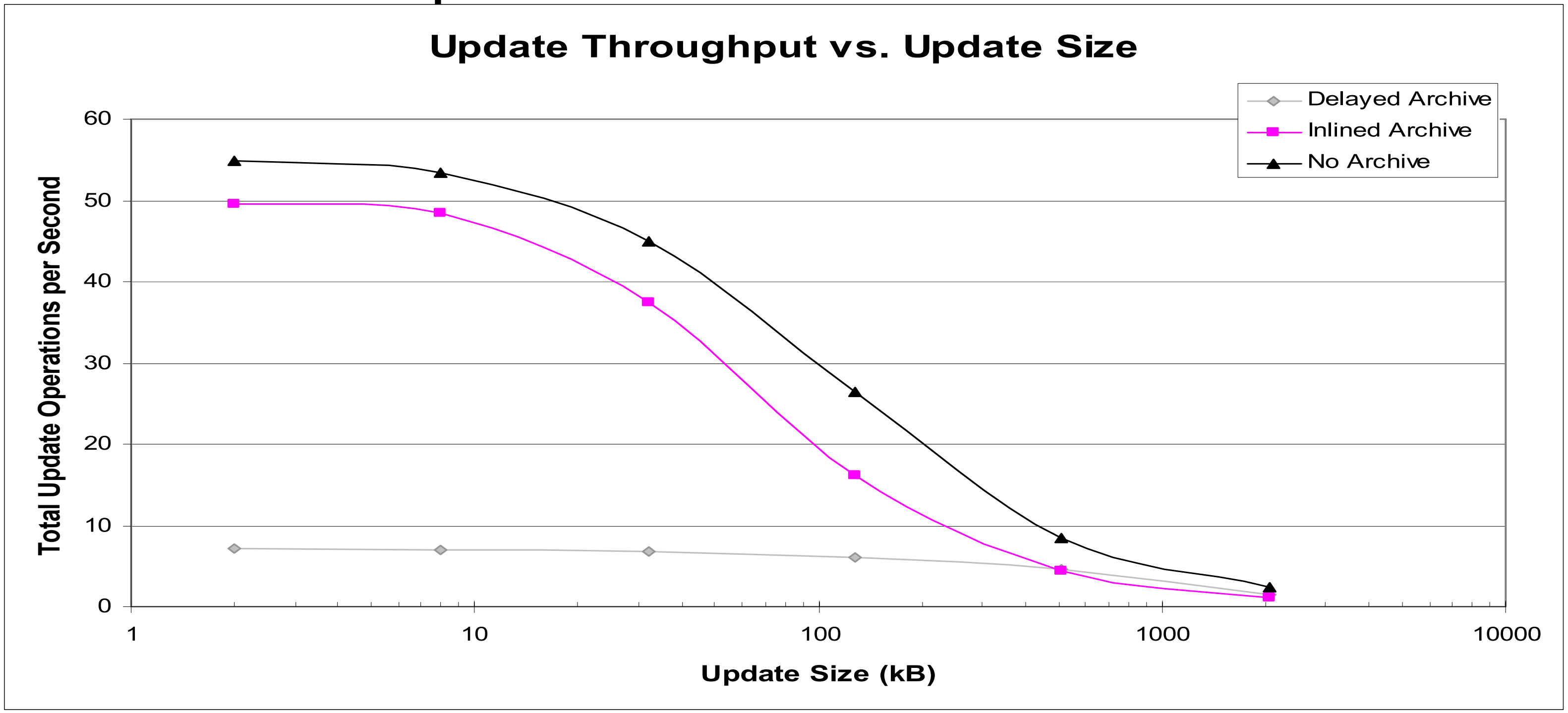
Performance: Throughput I

- Data Throughput
  - No archive 5MB/s.
  - Delayed 3MB/s.
  - Inlined 2.5MB/s.



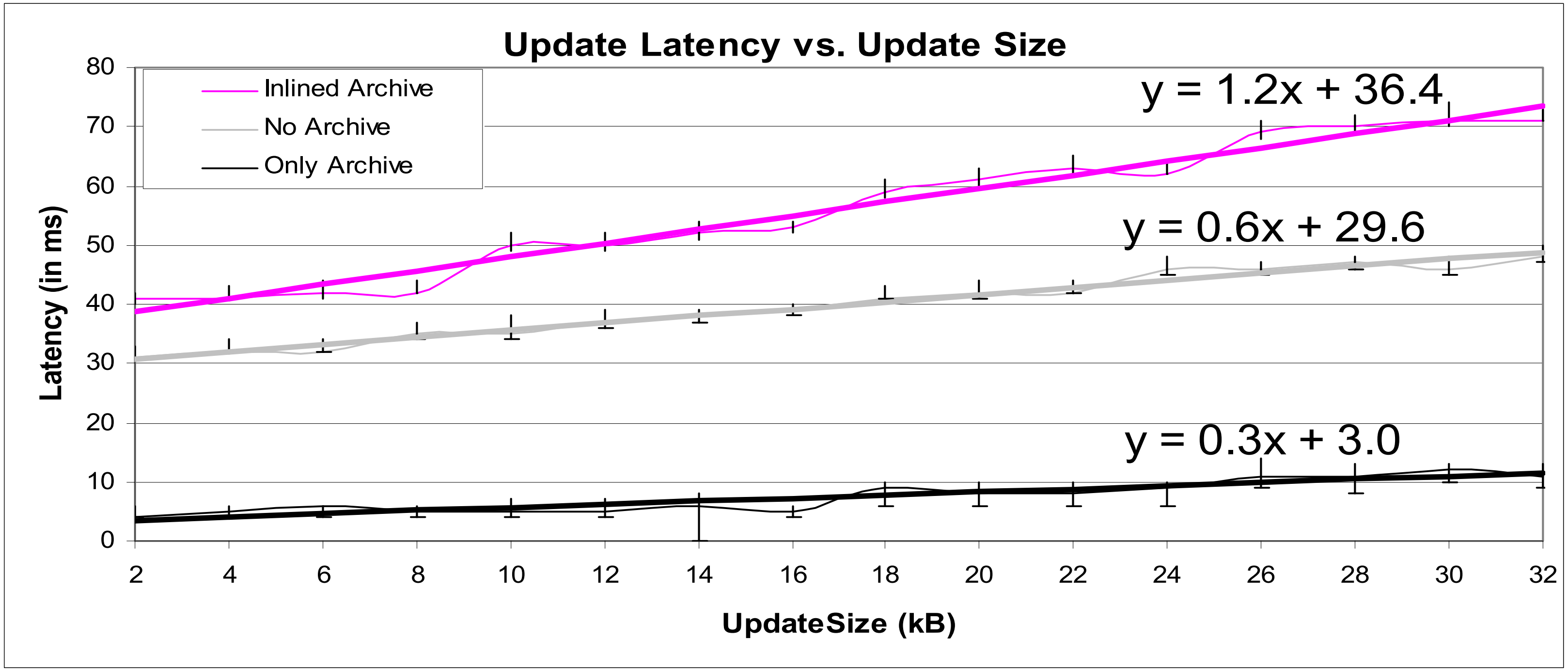
Performance: Throughput II

- Operations Throughput
  - No archive 55 opt/s. Small writes.
  - Delayed 8 opt/s. Small writes.
  - Inlined 50 opt/s. Small writes.



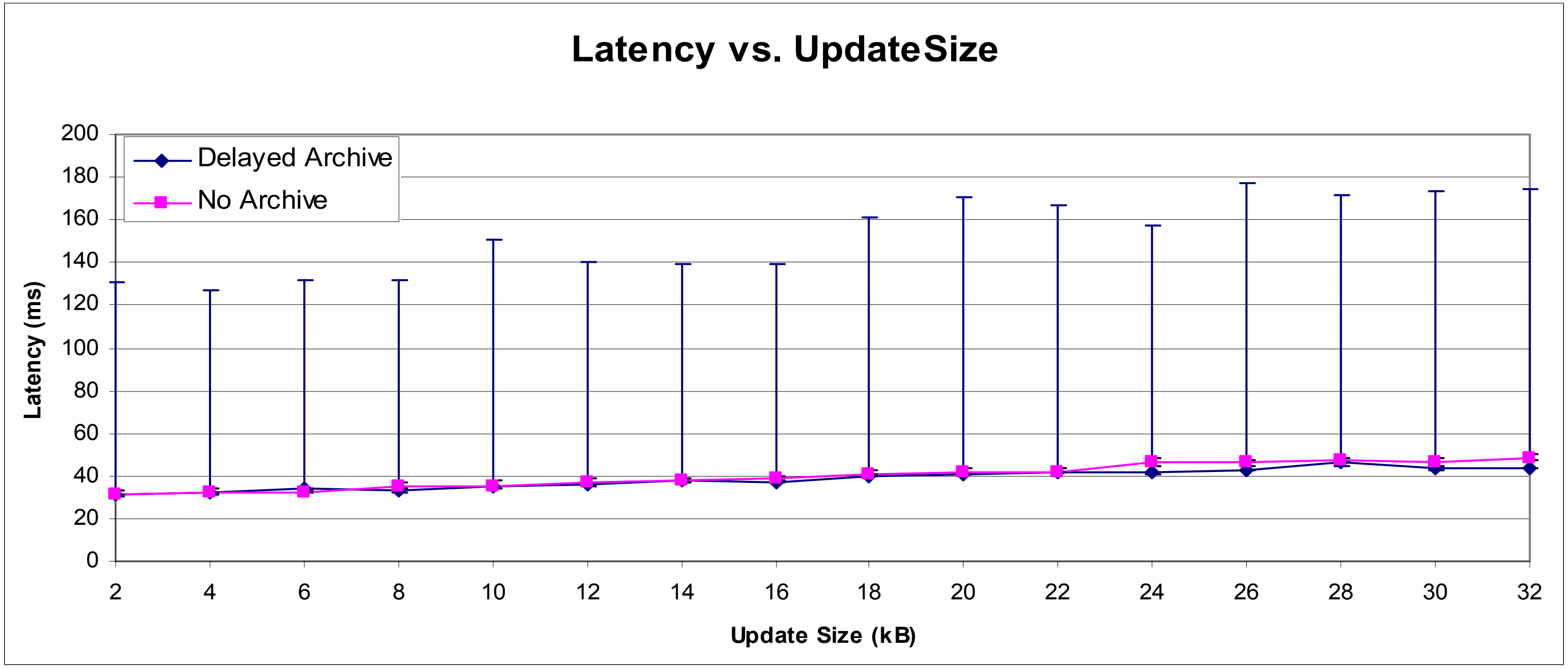
Performance: Latency I

- Latency
  - Archive only
    - Y-intercept 3ms, slope 0.3s/MB.
  - Inlined Archive = No archive + only archive.



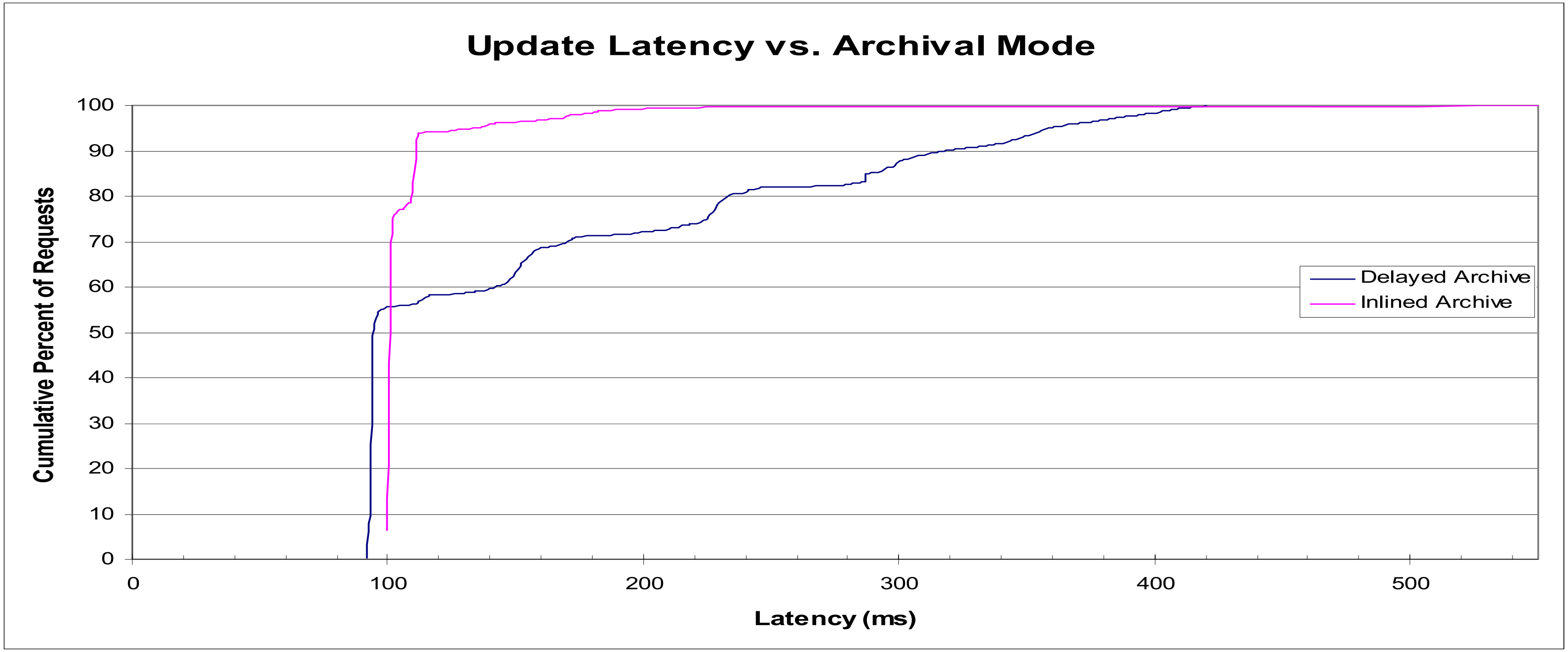
Performance: Latency II

- Latency
  - No Archive. Low variance.
  - Delayed Archive. High variance.



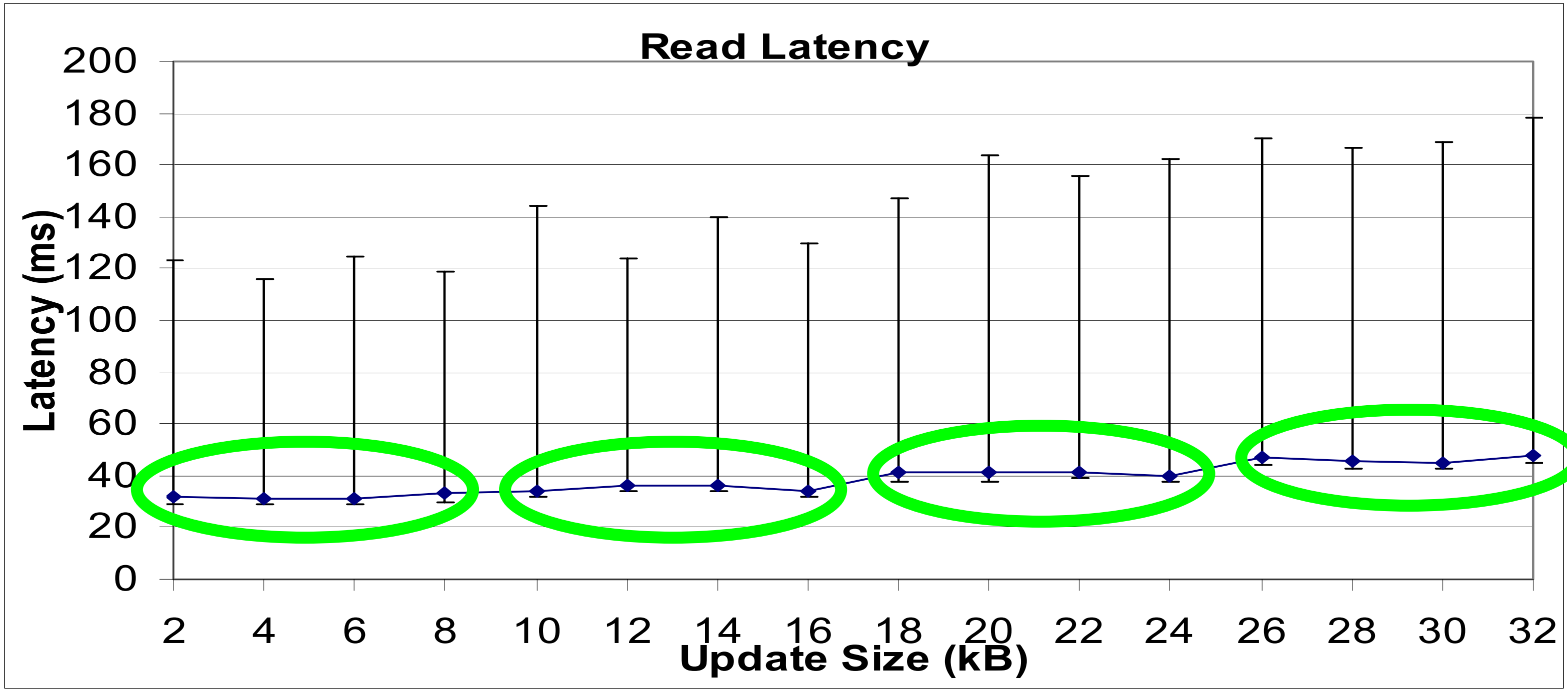
Performance: Latency III

- Latency
  - Inlined Archive. Low variance.
  - Delayed Archive. High variance.



Performance: Read Latency

- Read Latency
  - Low median.
  - High variance.
    - Queuing effect.



Future Directions

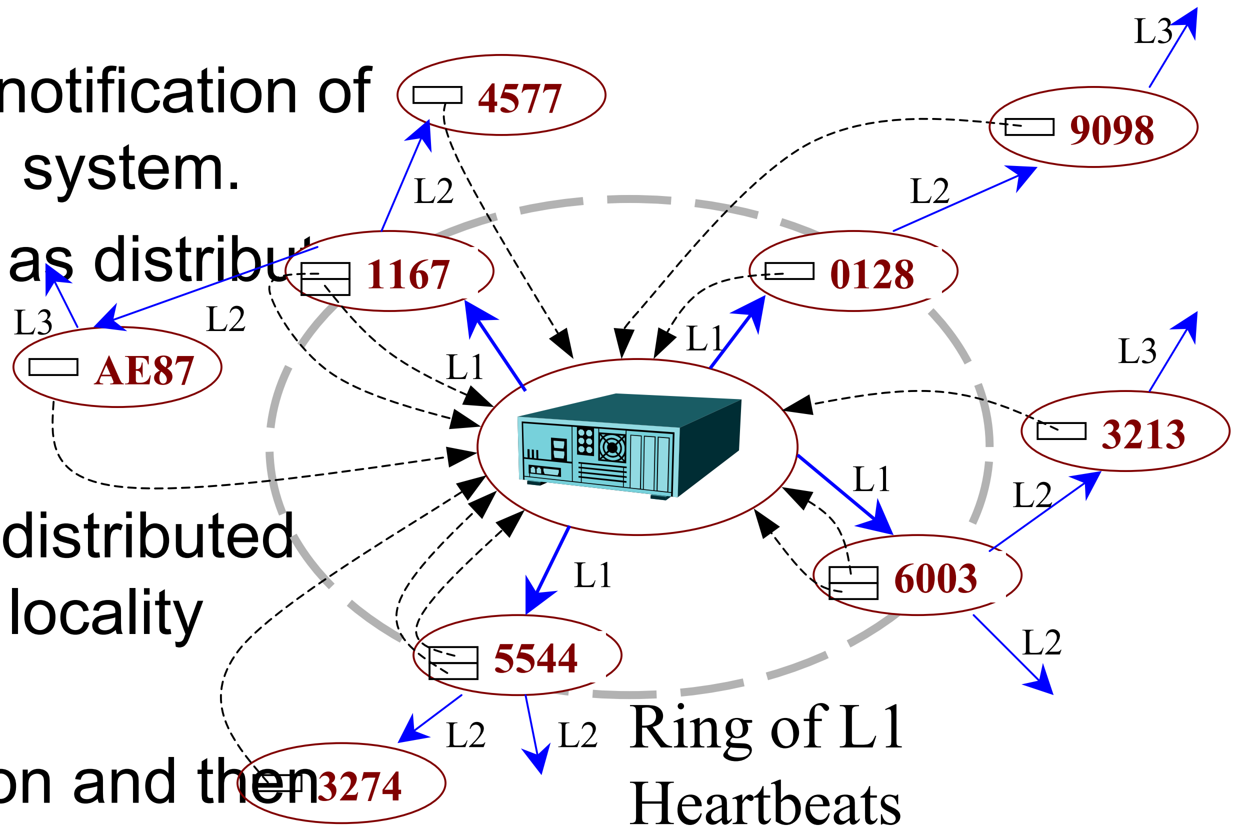
- Caching for performance
  - Automatic Replica Placement
- Automatic Repair
- Low Failure Correlation Dissemination

Caching

- Automatic Replica Placement
  - Replicas are *soft-state*.
  - Can be constructed and destroyed as necessary.
- Prefetching
  - Reconstruct replicas from fragments in advance of use

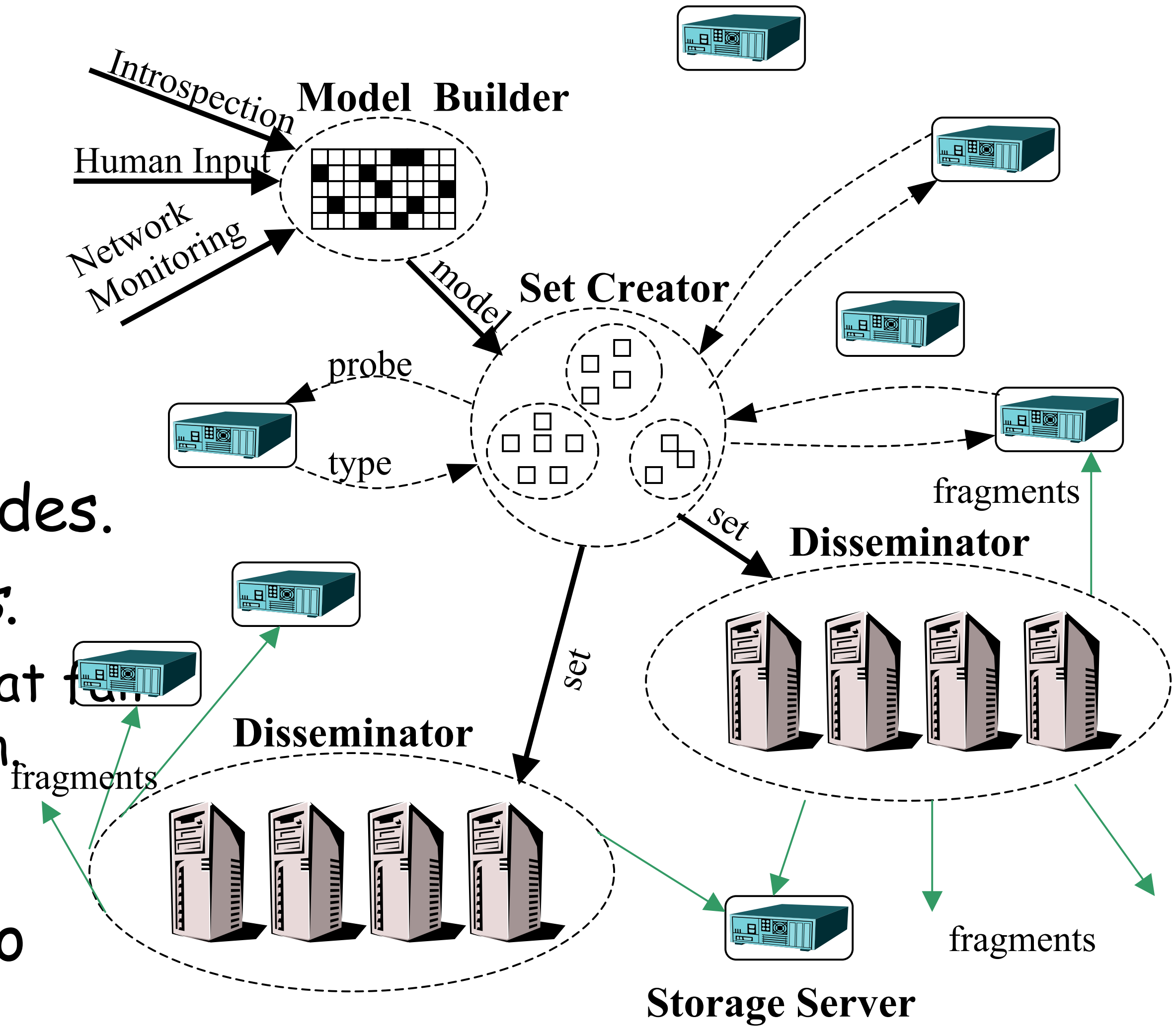
Efficient Repair

- Global.
  - Global Sweep and repair not efficient.
  - Want detection/notification of node removal in system.
  - Not as affective as distributed mechanisms.
- Distributed.
  - Exploit DOLR's distributed information and locality properties.
  - Efficient detection and then reconstruction of fragments.



Low Failure Correlation Dissemination

- Model Builder.
  - Various sources.
  - Model failure correlation.
- Set Creator.
  - Queries random nodes.
  - *Dissemination Sets.*
    - Storage servers that fail with low correlation.
- Disseminator.
  - Sends fragments to members of set.



Conclusion

- Storage efficient, self-verifying mechanism.
  - Erasure codes are good.
- Performance is good.
  - Improvements forthcoming.
- Self-verifying data assist in
  - Secure read-only data
  - Secure caching infrastructures
  - Continuous adaptation and repair

For more information:
http://oceanstore.cs.berkeley.edu/