Pond: the OceanStore Prototype

CS 6464

Cornell University

Presented by Yeounoh Chung

Motivation / Introduction

- “OceanStore is an Internet-scale, persistent dat a store”
- “for the first time, one can imagine providing t ruly durable, self maintaining storage to every computer user.”
- “vision” of highly available, reliable, and persi stent, data store utility model
  - Amazon S3 ?!

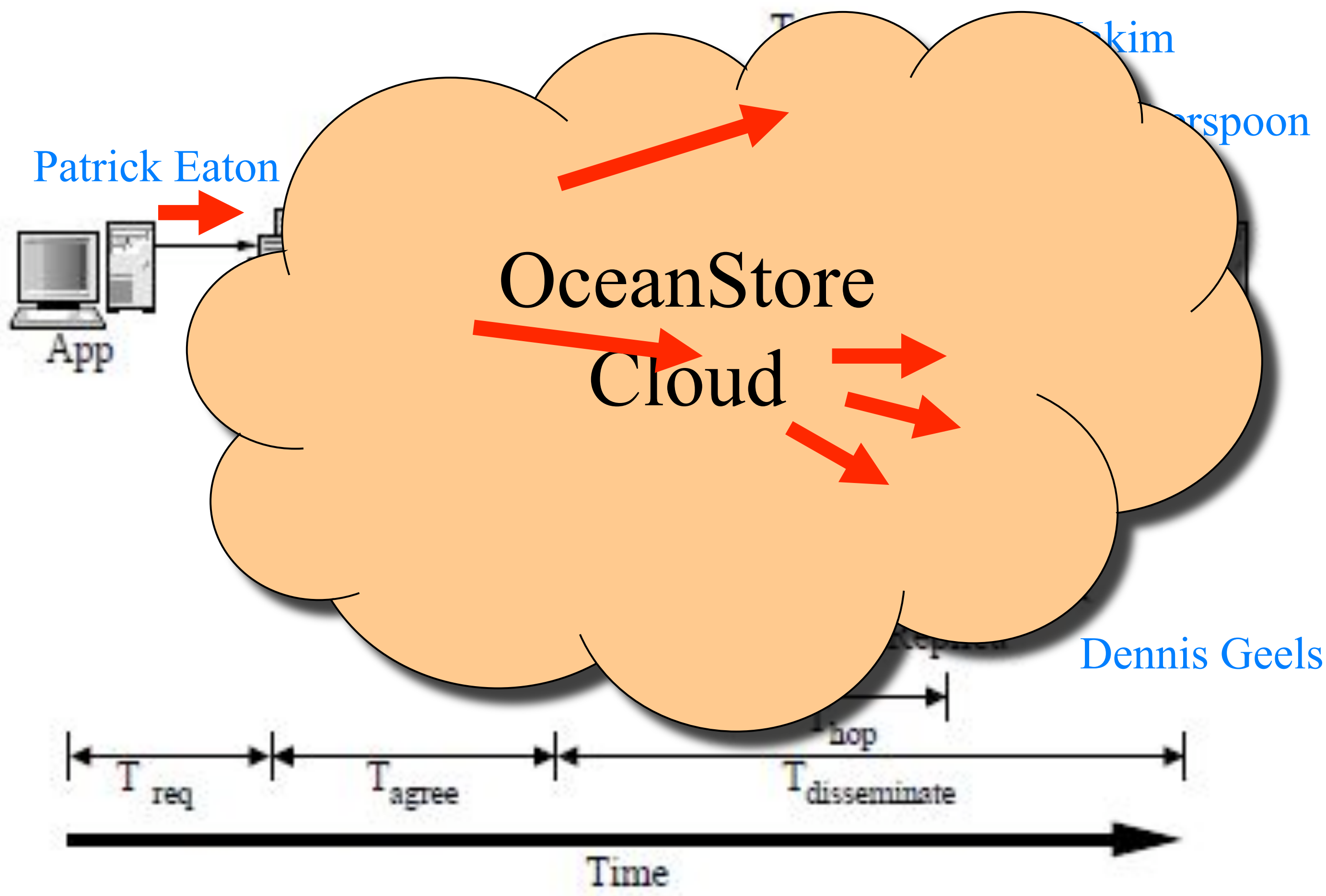
Motivation / Introduction

OceanStore (1999)	Amazon S3 (2009)
•Universal availability	•High availability
•High durability	•High durability
•Incremental Scalability	•High scalability
•Self-maintaining	•Self-maintaining
•Self-organizing	•Self-organizing
•Virtualization	•Virtualization
•Transparency	•Transparency
•Monthly access fee	•Pay-per-use fee
•Untrusted infrastructure	•Trusted infrastructure
•Two-tier system	•Single-tier system

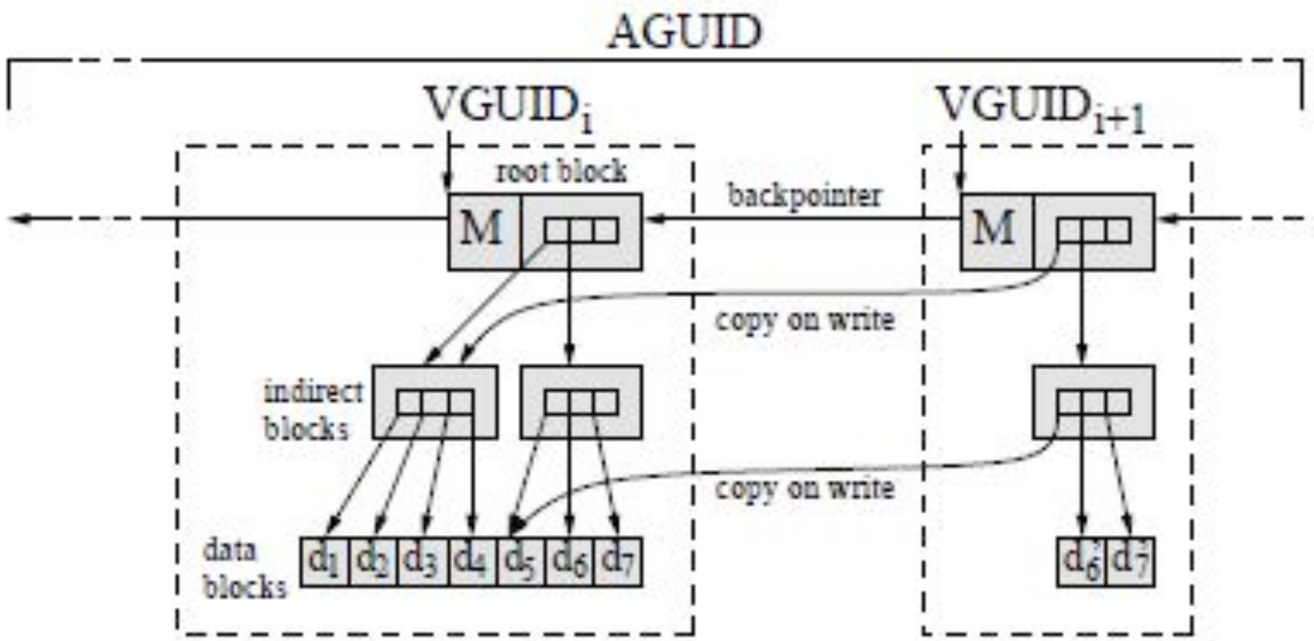
Outline

- Motivation / Introduction
- System Overview
- Consistency
- Persistency
- Failure Tolerance
- Implementation
- Performance
- Conclusion
- Related Work

System Overview



System Overview (Data Object)



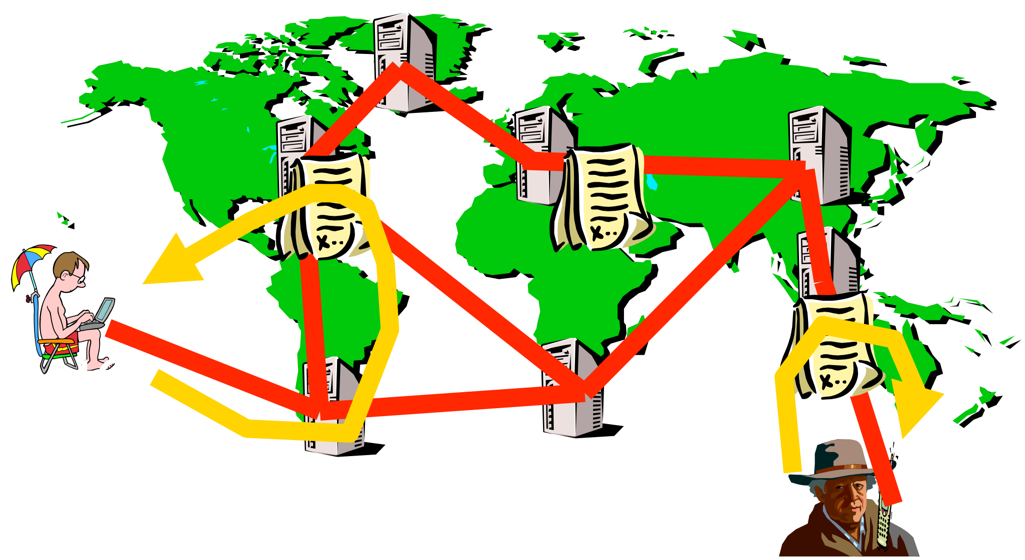
Name	Meaning	Description
BGUID	block GUID	secure hash of a block of data
VGUID	version GUID	BGUID of the root block of a version
AGUID	active GUID	names a complete stream of versions

System Overview (Update Model)

- Updates are applied atomically
- An array of actions guarded by a predicate
- No explicit locks
- Application-specific consistency
  - e.g. database, mailbox

System Overview (Tapestry)

- Scalable overlay network, built on TCP/IP
- Performs DOLR based on GUID
  - Virtualization
  - Location independence
- Locality aware
- Self-organizing
- self-maintaining

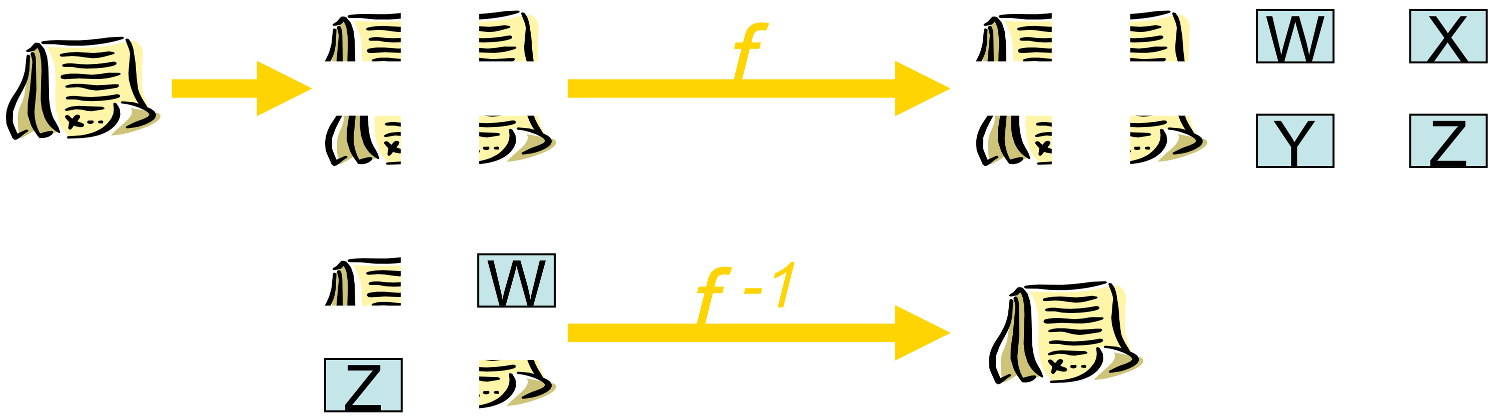


System Overview (Primary Rep.)

- Each data object is assigned an inner-ring
- Apply updates and create new versions
- Byzantine fault-tolerant
- Ability to change inner-ring servers any time
  - public key cryptography, proactive threshold signature, Tapestry
- Responsible party

System Overview (Achi. Storage)

- Erasure codes are more space efficient

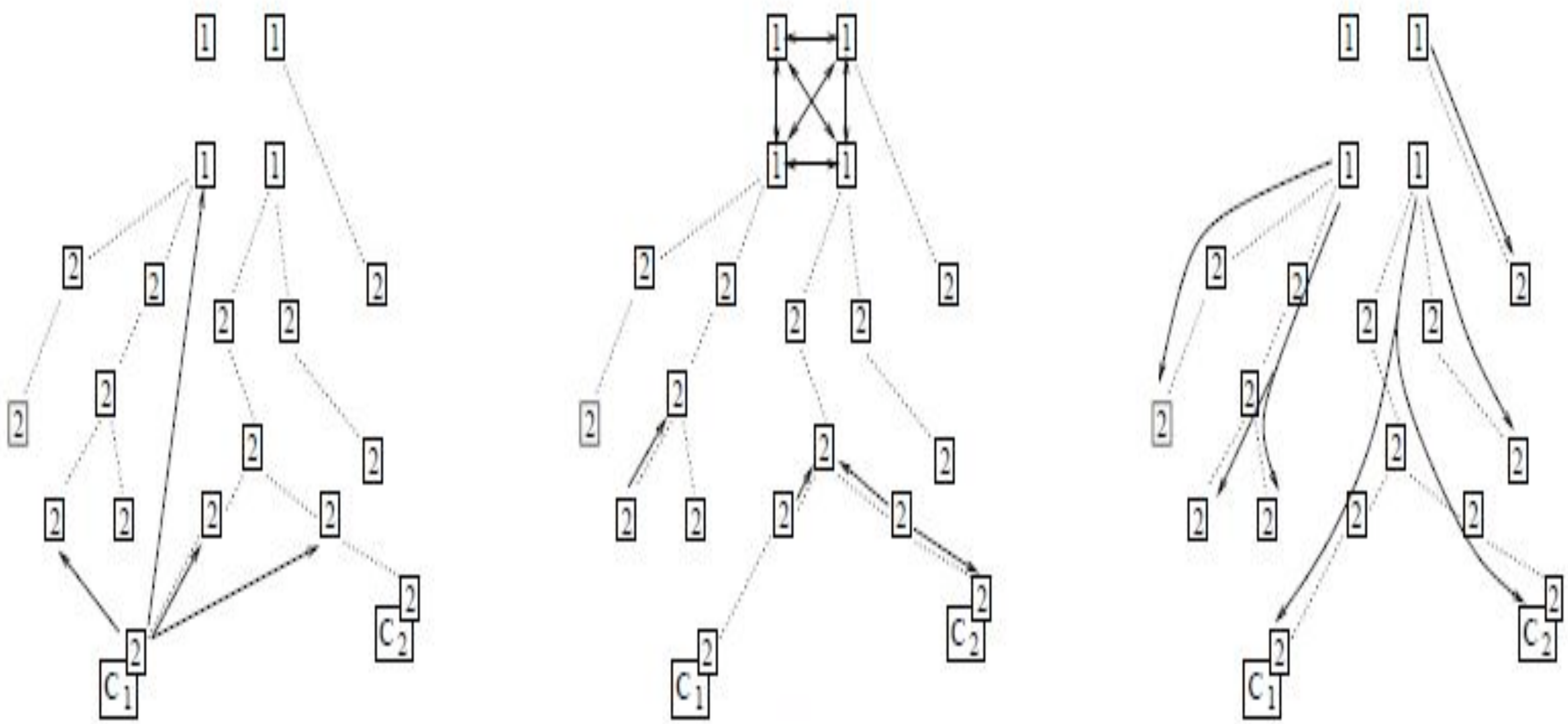


- Fragments are distributed uniformly among archival storage servers
  - BGUID, n_frag
- Pond uses Cauchy Reed-Solomon code

System Overview (Caching)

- Promiscuous caching
- Whole-block caching
- Host caches the read block and publishes its po session in Tapestry
- Pond uses LRU
- Use Heartbeat to get the most recent copy

System Overview (Diss. Tree)



(a) (b) (c)

Outline

- Motivation / Introduction
- System Overview
- Consistency
- Persistency
- Failure Tolerance
- Implementation
- Performance
- Conclusion
- Related Work

Consistency (Primary Replica)

- Read-only blocks
- Application-specific consistency
- Primary-copy replication
  - heartbeat <AGUID, VGUID, t, v_seq>

Persistency (Archival Storage)

- Archival storage
- Aggressive replication
- Monitoring
  - Introspection
- Replacement
  - Tapestry

Failure Tolerance (Everybody)

- All newly created blocks are encoded and stored in Archival servers
- Aggressive replication
- Byzantine agreement protocol for inner-ring
- Responsible Party
  - single point of failure?
  - scalability?

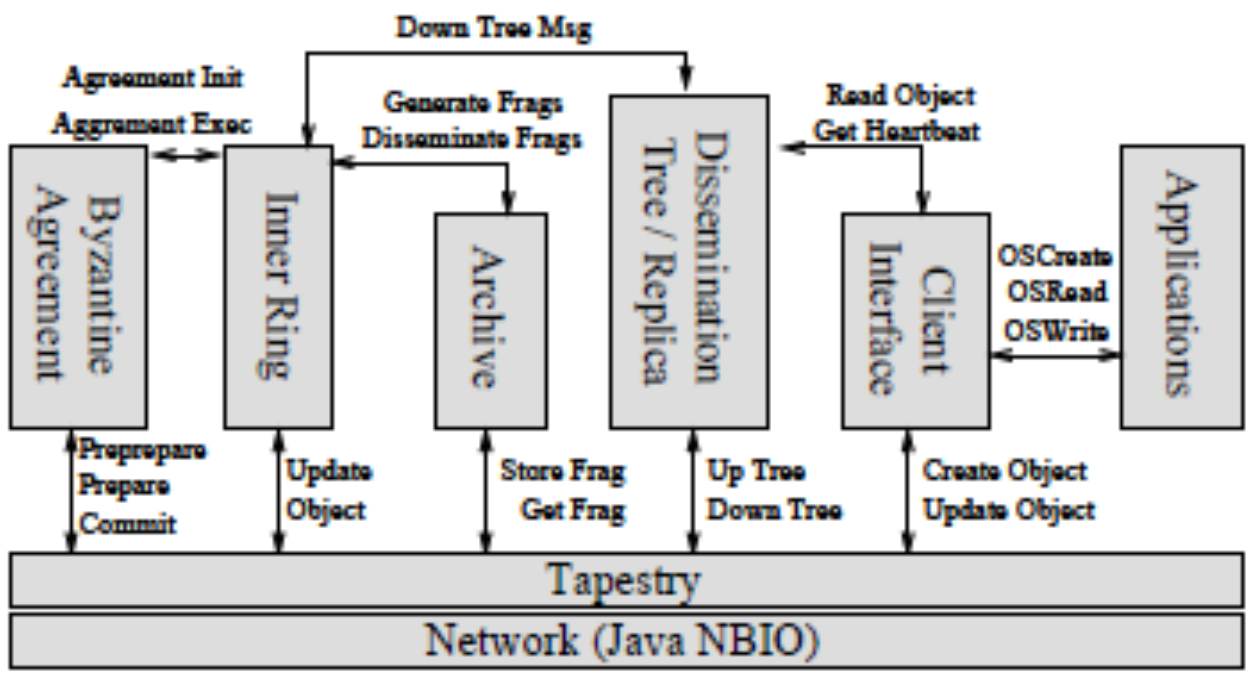
Outline

- Motivation / Introduction
- System Overview
- Consistency
- Persistency
- Failure Tolerance
- Implementation
- Performance
- Conclusion
- Related Work

Implementation

- Built in Java, atop SEDA
- Major subsystems are functional
  - self-organizing Tapestry
  - primary replica with Byzantine agreement
  - self-organizing dissemination tree
  - erasure-coding archive
  - application interface: NFS, IMAP/SMTP, HTTP

Implementation



Outline

- Motivation / Introduction
- System Overview
- Consistency
- Persistency
- Failure Tolerance
- Implementation
- Performance
- Conclusion
- Related Work

Performance

- Updata performance
- Dissemination tree performance
- Archival retrieval performance
- The Andrew Benchmark

Performance (test beds)

- Local cluster
  - 42 machines at Berkeley
  - 2x 1.0 GHz CPU, 1.5 GB SDRAM, 2x 36 GB hard drives
  - gigabit Ethernet adaptor and switch
- PlanetLab
  - 101 nodes across 43 sites
  - 1.2 GHz, 1 GB memory

Performance (update)

Key Size	Update Size	Archive	Update Latency (ms) 5%	Median	95%
512	4 kB	off	36	37	38
		on	39	40	41
	2 MB	off	494	513	778
		on	1037	1086	1348
1024	4 kB	off	94	95	96
		on	98	99	100
	2 MB	off	557	572	875
		on	1098	1150	1448

Table 2: *Results of the Latency Microbenchmark in the Local Area.* All nodes are hosted on the cluster. Ping latency between nodes in the cluster is 0.2 ms. We run with the archive enabled and disabled while varying the update size and key length.

Phase	Time (ms) 4 kB Update	2 MB Update
Check Validity	0.3	0.4
Serialize	6.1	26.6
Update	1.5	113.0
Archive	4.5	566.9
Sign Result	77.8	75.8

Table 3: *Latency Breakdown of an Update.* The majority of the time in a small update performed on the cluster is spent computing the threshold signature share over the result. With larger updates, the time to apply and archive the update dominates signature time.

Performance (update)

Inner Ring	Client	Avg. Ping	Update Size	Update Latency (ms) 5%	Median	95%
Cluster	Cluster	0.2	4 kB	98	99	100
			2 MB	1098	1150	1448
Cluster	UCSD	27.0	4 kB	125	126	128
			2 MB	2748	2800	3036
Bay Area	UCSD	23.2	4 kB	144	155	166
			2 MB	8763	9626	10231

Table 4: *Results of the Latency Microbenchmark Run in the Wide Area.* All tests were run with the archive enabled using 1024-bit keys. "Avg. Ping" is the average ping time in milliseconds from the client machine to each of the inner ring servers. UCSD is the University of California at San Diego.

IR Location	Client Location	Throughput (MB/s)
Cluster	Cluster	2.59
Cluster	PlanetLab	1.22
Bay Area	PlanetLab	1.19

Table 5: *Throughput in the Wide Area.* The throughput for a distributed ring is limited by the wide-area bandwidth. All tests are run with the archive on and 1024-bit keys.

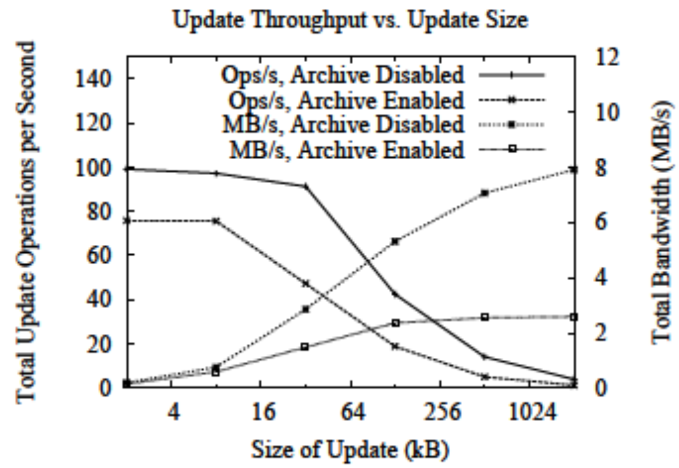


Figure 5: *Throughput in the Local Area.* This graph shows the update throughput in terms of both operations per second (left axis) and bytes per second (right axis) as a function of update size. While the ops/s number falls off quickly with update size, throughput in bytes per second continues to increase. All experiments are run with 1024-bit keys. The data shown is the average of three trials, and the standard deviation for all points is less than 3% of the mean.

Performance (archival)

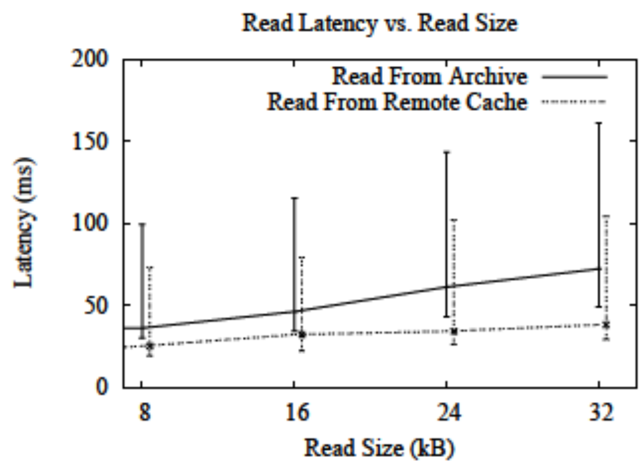


Figure 6: *Latency to Read Objects from the Archive.* The latency to read data from the archive depends on the latency to retrieve enough fragments for reconstruction.

Performance (dissemination tree)

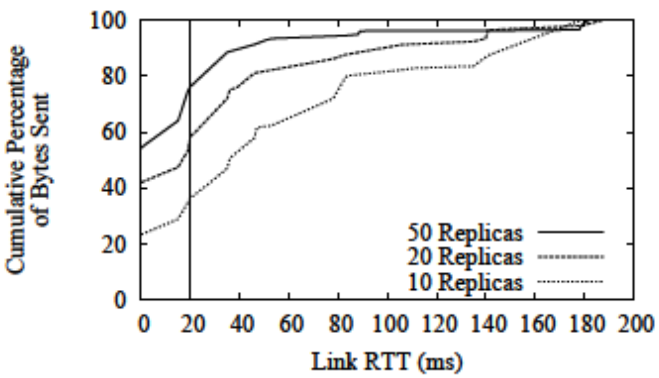


Figure 7: *Results of the Stream Benchmark.* The graph shows the percentage of bytes sent over links of different latency as the number of replicas varies.

Tokens Passed Using	Latency per Tag (ms)
OceanStore	329
Tapestry	104
TCP/IP	73

Table 6: *Results of the Tag Microbenchmark.* Each experiment was run at least three times, and the standard deviation across experiments was less than 10% of the mean. All experiments are run using 1024-bit keys and with the archive disabled.

Performance (Andrew benchmark)

	LAN			WAN		
	Linux	OceanStore		Linux	OceanStore	
Phase	NFS	512	1024	NFS	512	1024
I	0.0	1.9	4.3	0.9	2.8	6.6
II	0.3	11.0	24.0	9.4	16.8	40.4
III	1.1	1.8	1.9	8.3	1.8	1.9
IV	0.5	1.5	1.6	6.9	1.5	1.5
V	2.6	21.0	42.2	21.5	32.0	70.0
Total	4.5	37.2	73.9	47.0	54.9	120.3

Table 7: *Results of the Andrew Benchmark.* All experiments are run with the archive disabled using 512 or 1024-bit keys, as indicated by the column headers. Times are in seconds, and each data point is an average over at least three trials. The standard deviation for all points was less than 7.5% of the mean.

Conclusion

- Pond is a working subset of the vision
- Promising in WAN
- Threshold signatures, erasure-coded archival are expensive
- Pond is fault tolerant system, but it is not tested with any failed node
- Any thoughts?

Related Work

- FarSite
- ITTC, COCA
- PAST, CFS, IVY
- Pangaea