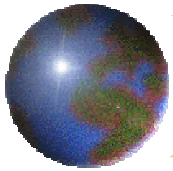
Pond

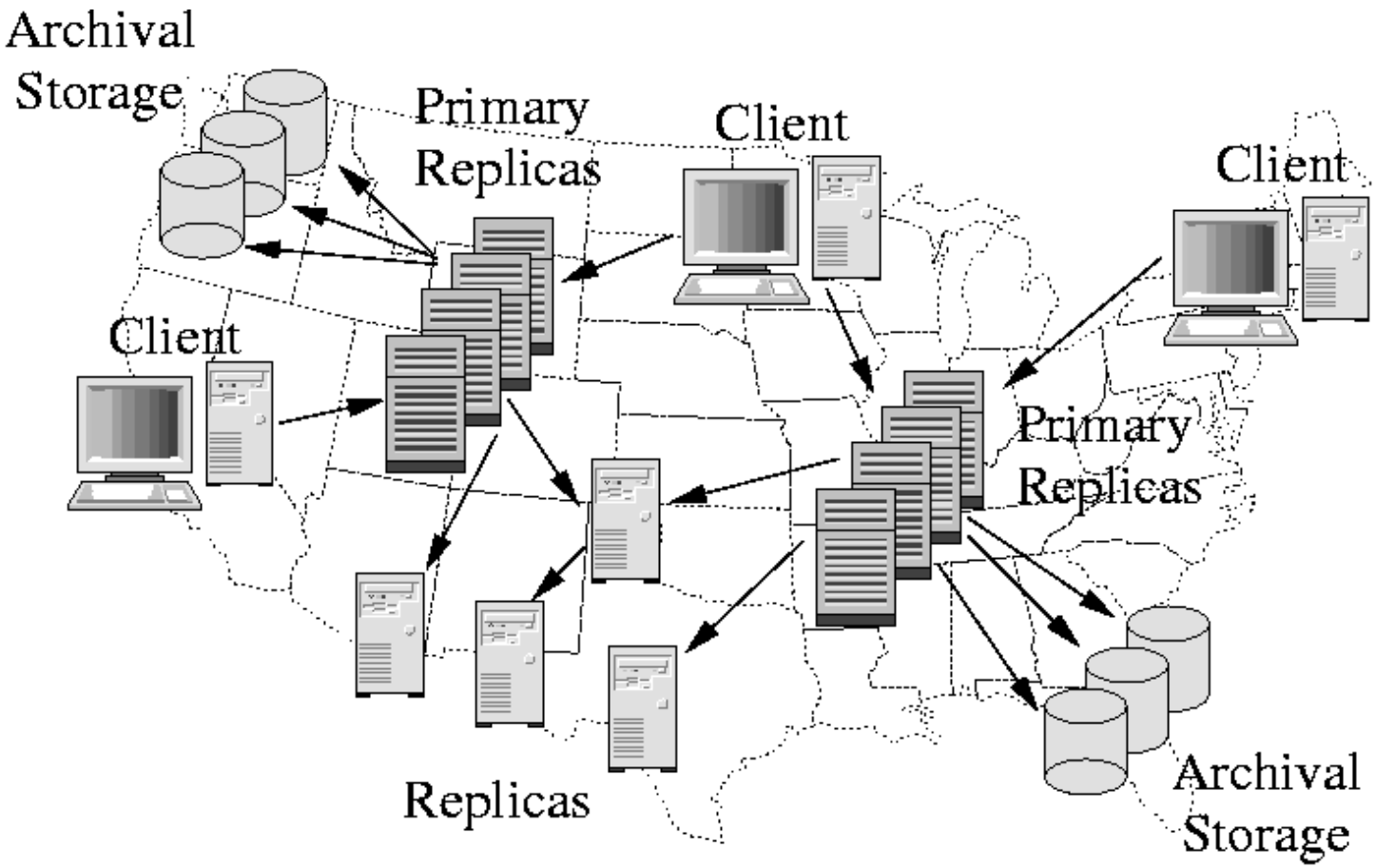
The OceanStore Prototype



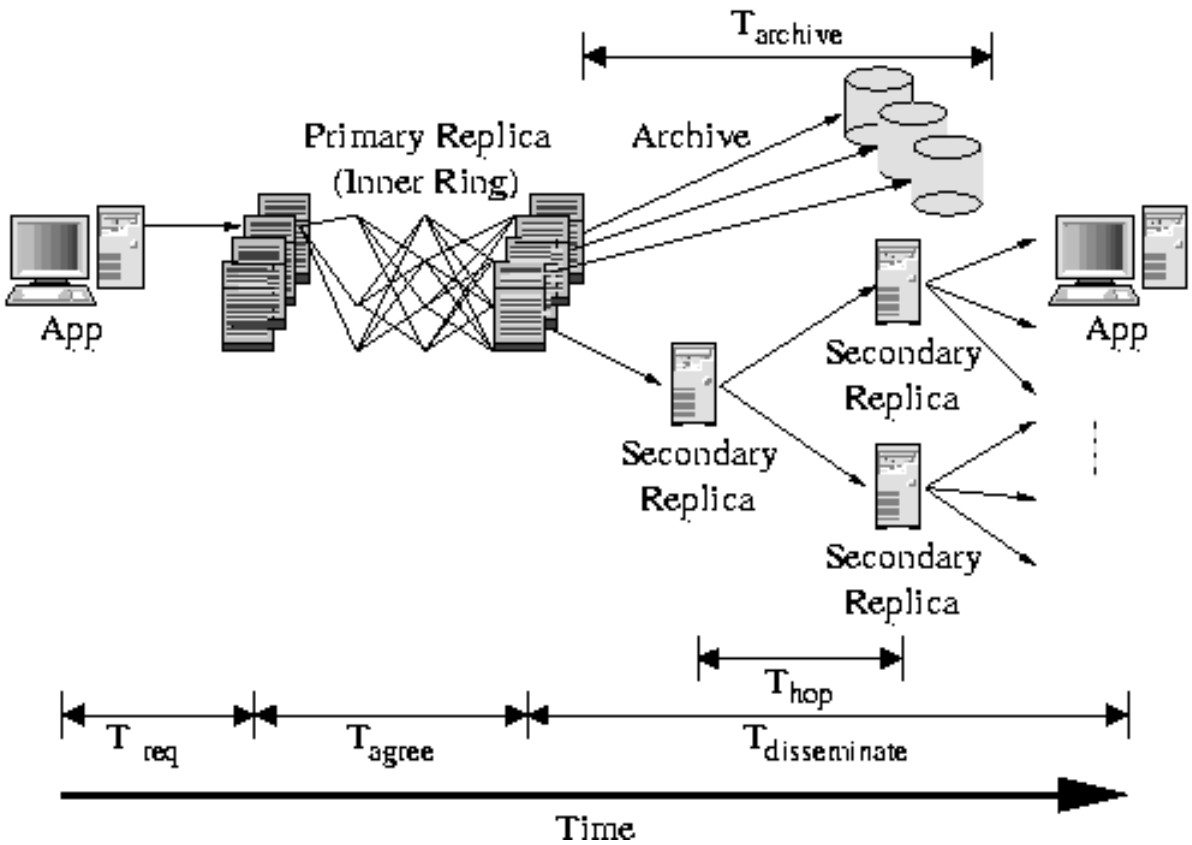
Talk Outline

- System overview
- Implementation status
- Results from FAST paper
- Conclusion

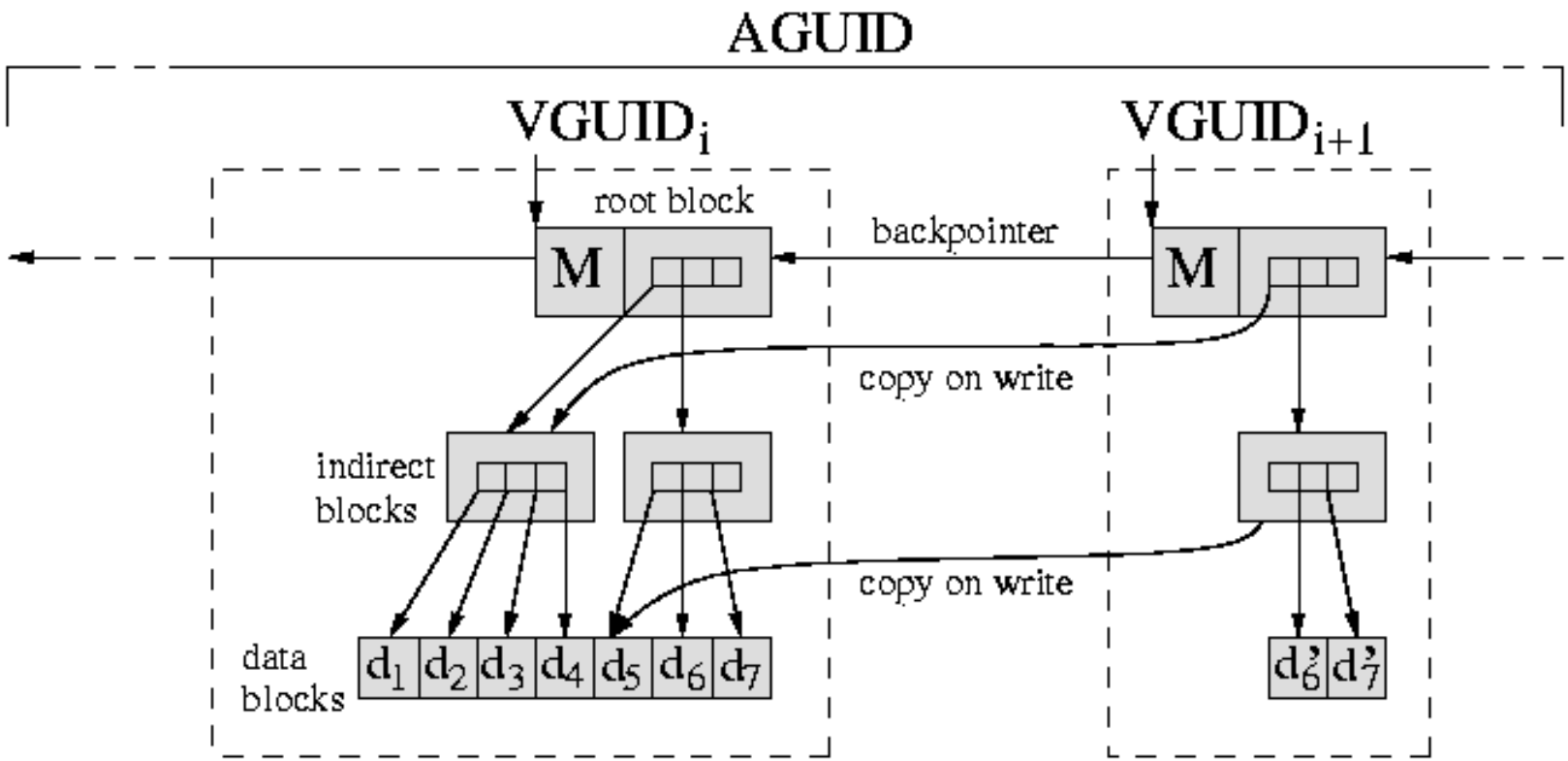
OceanStore System Layout

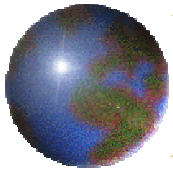


The Path of an Update



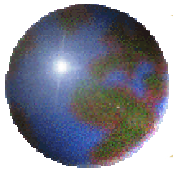
Data Object Structure





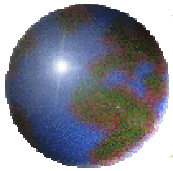
Talk Outline

- System overview
- Implementation status
- Results from FAST paper
- Conclusion



Prototype Implementation

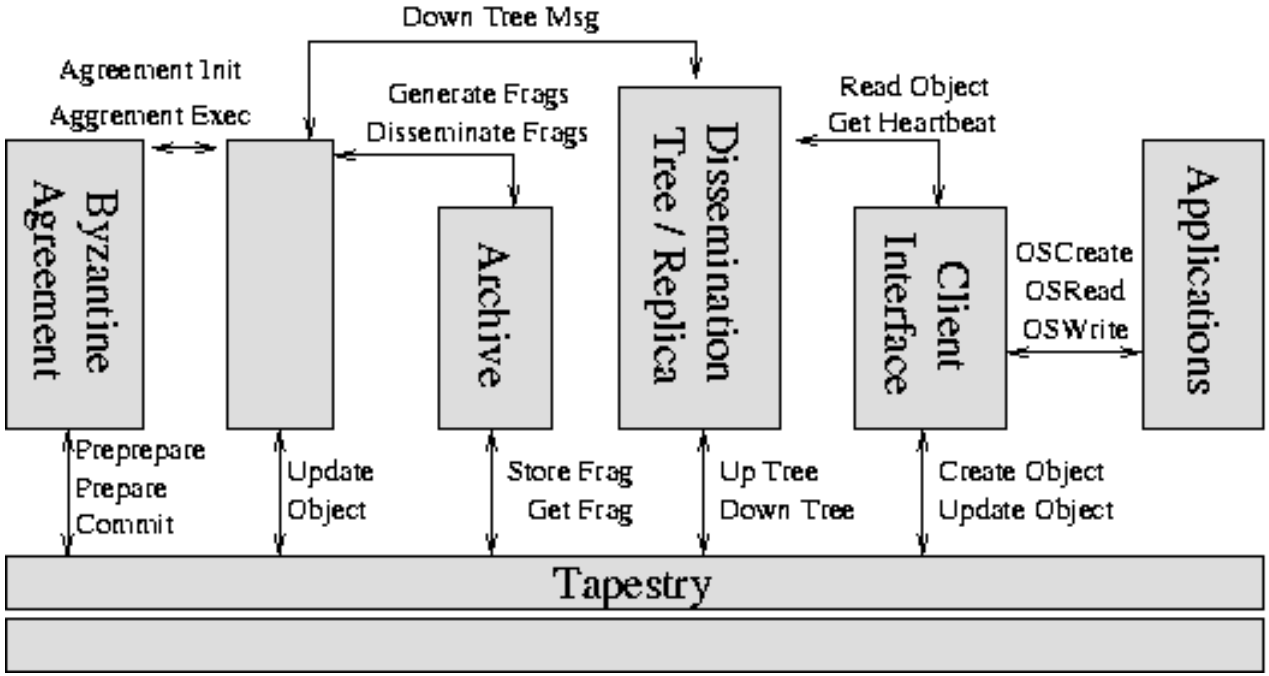
- All major subsystems operational
  - Fault-tolerant inner ring
  - Self-organizing second tier
  - Erasure-coding archive
  - Multiple application interfaces: NFS, IMAP/SMTP, HTTP



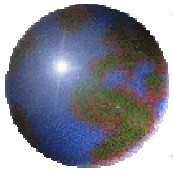
Prototype Implementation

- Missing pieces
  - Full Byzantine-fault-tolerant agreement
  - Tentative update sharing
  - Inner ring membership rotation
  - Flexible ACL support
  - Proactive replica placement

Software Architecture

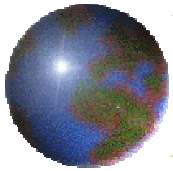


- 20 SEDA stages
- 280K Lines of Java (J2SE v1.3)
- JNI libraries for crypto, archive



Running OceanStore

- Host machines must have JRE
  - x86 libraries provided
- Upload package, SSH public keys
  - ~4MB
- Centralized control: run-experiment
  - Builds, ships per-host configuration
  - Starts remote processes
  - Scans logs for completion or errors
  - Support for virtual nodes



Example configuration

System description

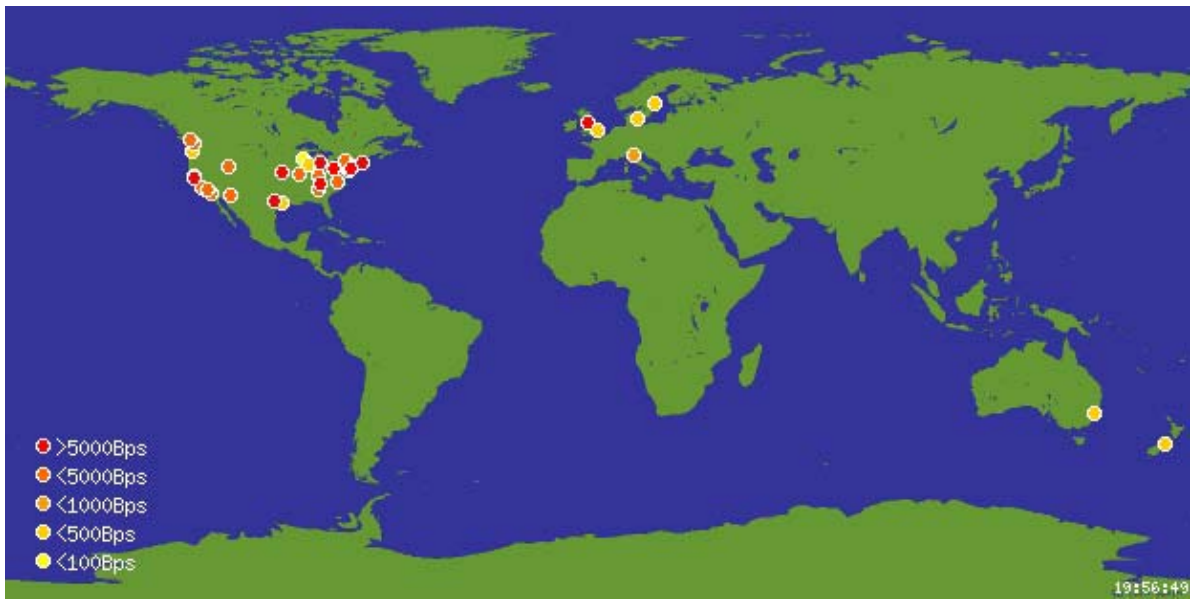
hosts monkey.cs orangutan.cs ….

```
ULNFS    ulnfs.cfg   dynamic  mortal 0
RP       rp.cfg        static  daemon  0
Ring0    inner.cfg     static  daemon  1
Ring1    inner.cfg     static  daemon  2
Ring2    inner.cfg     static  daemon  3
Ring3    inner.cfg     static  daemon  4
Archive0 storage.cfg  static  daemon  5
Archive1 storage.cfg  static  daemon  5
Archive2 storage.cfg  static  daemon  5
Archive3 storage.cfg  static  daemon  6
Archive4 storage.cfg  static  daemon  6
         ….
```

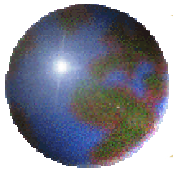
Node template

```
<sandstorm>
  <!include Generic.hdr>
  <stages>
    <!include Network.stg>
    <RpcStage>
          class ostore.apps.ulnfs.RpcStage
          <initargs>
            mountd_port      2635
            nfsd_port        3049
            node_id          ${NodeID}
          </initargs>
    </RpcStage>
    <!include Client.stg>
                                    ….
```

Deployment: PlanetLab



- http://www.planet-lab.org
- ~100 hosts, ~40 sites
- Pond: up to 1000 virtual nodes
- 5 minute startup

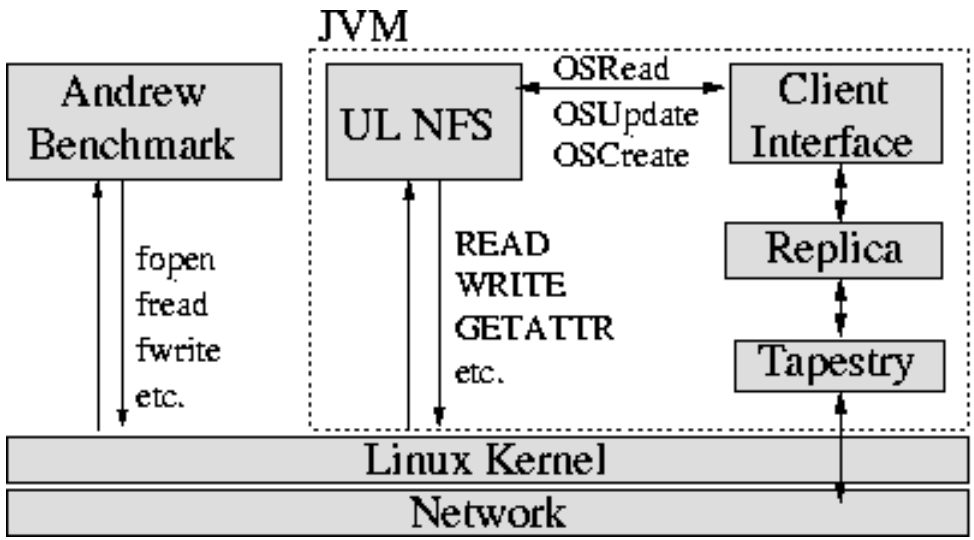


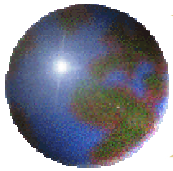
Talk Outline

- System overview
- Implementation status
- Results from FAST paper
- Conclusion

Results: Andrew Benchmark

- Ran MAB on Pond using User Level NFS (ULNFS)
  - Strong consistency restrictions for directories
  - Loose consistency for files allows caching, interleaved writes
  - Benefits: Security, Durability, Time travel, etc.

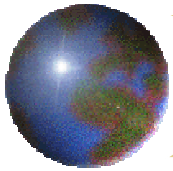




Results: Andrew Benchmark

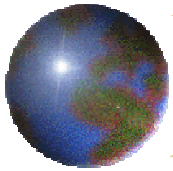
Phase	Linux	Pond-512	Pond-1024
I	0.9	2.8	6.6
II	9.4	16.8	40.4
III	8.3	1.8	1.9
IV	6.9	1.5	1.5
V	21.5	32.0	70.0
Total	47.0	54.9	120.3

- 4.6x than NFS in read-intensive phases
- 7.3x slower in write-intensive phases



Closer look: Update Latency

- Inner Ring update algorithm:
  - All-pairs communication to agree to start
  - Each replica applies update locally
  - All-pairs to agree on result
  - Each replica signs certificate
    - Threshold Signature
- Robust to Byzantine failures of up to 1/3 of primary replicas



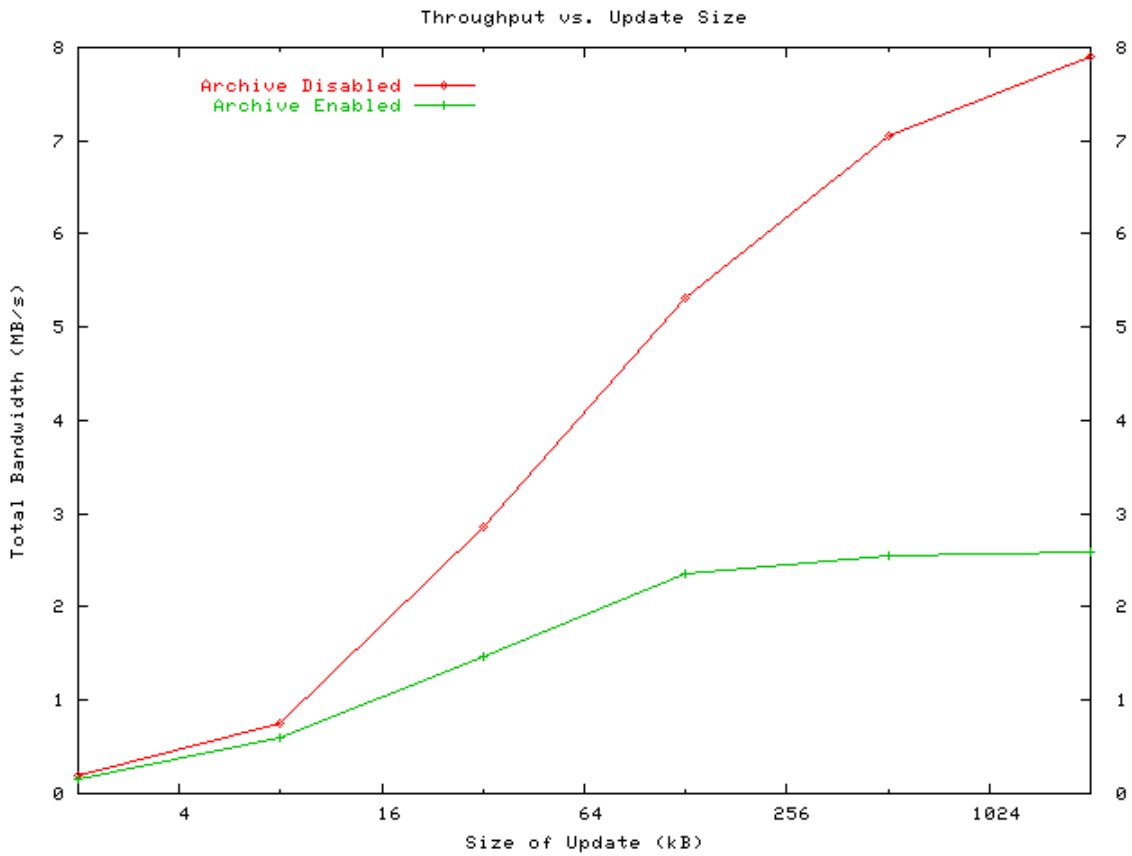
Closer look: Update Latency

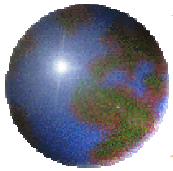
Update Latency (ms)				
Key Size	Update Size	5% Time	Median Time	95% Time
512b	4kB	39	40	41
	2MB	1037	1086	1348
1024b	4kB	98	99	100
	2MB	1098	1150	1448

Latency Breakdown	
Phase	Time (ms)
Check	0.3
Serialize	6.1
Apply	1.5
Archive	4.5
Sign	77.8

- Threshold Signature dominates small update latency
  - Common RSA tricks not applicable
- Batch updates to amortize signature cost
- Tentative updates hide latency

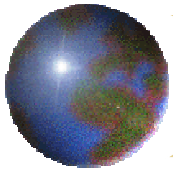
Closer Look: Update Throughput





Closer look: Dissemination Tree

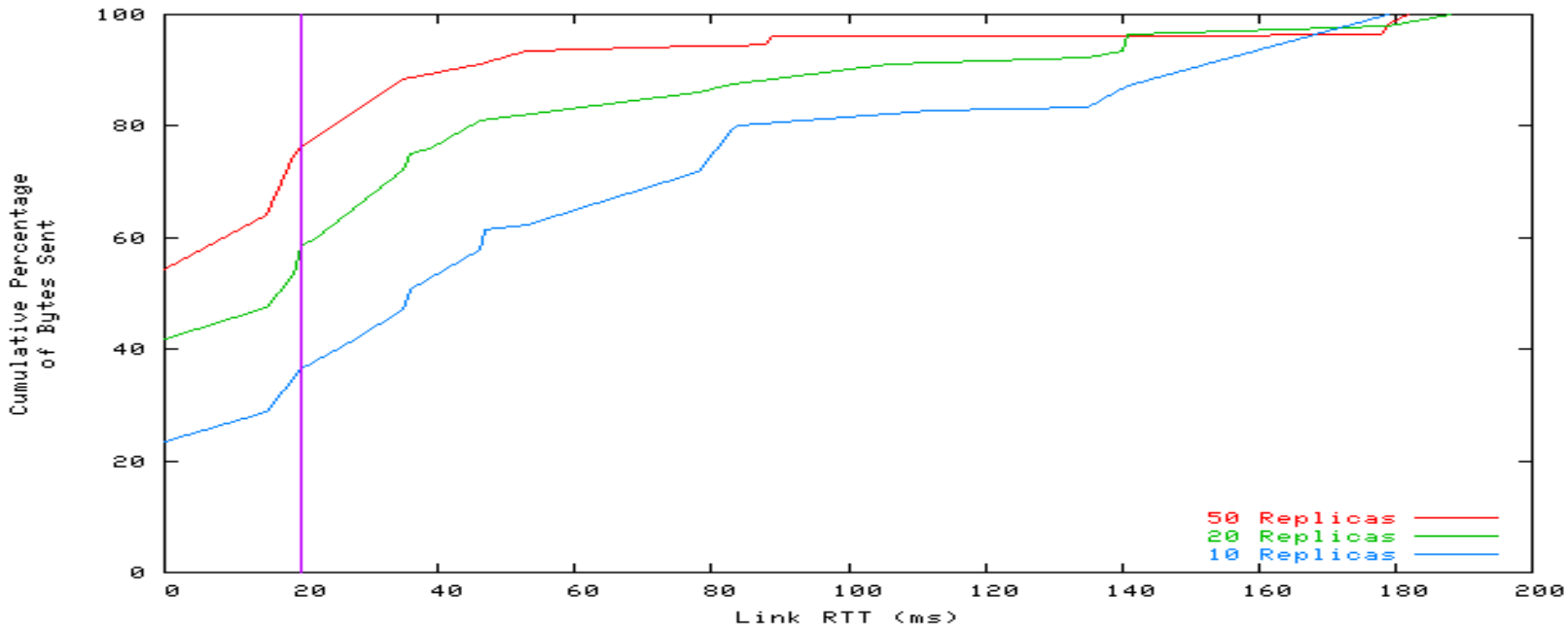
- Secondary replicas self-organize into application-level multicast tree
  - Shield inner ring from request load
  - Save bandwidth on update propagation
- Tree joining heuristic:
  - Connect to closest replica using Tapestry
  - Should minimize use of long-distance links



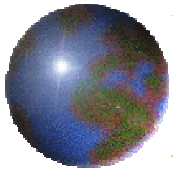
Stream Microbenchmark

- Designed to measure efficiency of dissemination tree
- Ran 500 virtual nodes on PlanetLab
  - Inner Ring in SF Bay Area
  - Replicas clustered in 7 largest P-Lab sites
- Streams updates to all replicas
  - One writer - *content creator* – repeatedly appends to data object
  - Others read new versions as they arrive
  - Measure network resource consumption

Results: Stream Microbenchmark

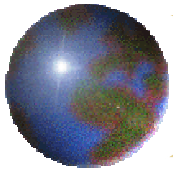


- Dissemination tree uses network resources efficiently
  - Most bytes sent across local links as second tier grows
- Acceptable latency increase over broadcast (33%)



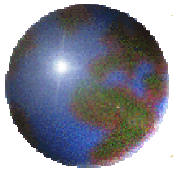
Talk Outline

- System overview
- Implementation status
- Results from FAST paper
- Conclusion



Conclusion

- Operational OceanStore Prototype
- Current Research Directions
  - Examine bottlenecks
  - Improve stability
  - Data Structure Improvement
  - Replica Management
  - Archival Repair



Availability

- FAST paper
  - "Pond: the OceanStore Prototype"
- More information
  - http://oceanstore.cs.berkeley.edu
  - http://oceanstore.sourceforge.net
- Demonstrations Available