Improving Bandwidth Efficiency of Peer-to-Peer Storage

Patrick Eaton, Emil Ong, John Kubiatowicz
University of California, Berkeley
http://oceanstore.cs.berkeley.edu/
P2P Storage: Promise vs. Reality
P2P Storage: Promise vs. Reality

• Promise: P2P storage is...
  – Globally accessible
  – Highly available
  – Extremely durable
P2P Storage: Promise vs. Reality

• Promise: P2P storage is...
  – Globally accessible
  – Highly available
  – Extremely durable
P2P Storage: Promise vs. Reality

• Promise: P2P storage is...
  – Globally accessible
  – Highly available
  – Extremely durable

• Reality: Many challenges…
Reality: Non-Uniform Network Connectivity

- Well-connected network core
- Weakly-connected clients
- Challenge: Reasonable performance for weakly-connected clients
Reality: Non-Uniform Network Connectivity

- Well-connected network core
- Weakly-connected clients
- Challenge: Reasonable performance for weakly-connected clients
Infrastructure Assumptions

- **Infrastructure helps manage objects**
  - Like FARSITE or OceanStore
  - Provides update application, serialization, durability, etc.
- **Partially trusted servers in network core**
  - Trusted to execute protocol, not trusted with user data
Infrastructure Assumptions

- Infrastructure helps manage objects
  - Like FARSITE or OceanStore
  - Provides update application, serialization, durability, etc.
- Partially trusted servers in network core
  - Trusted to execute protocol, not trusted with user data
Infrastructure Assumptions

- **Infrastructure helps manage objects**
  - Like FARSITE or OceanStore
  - Provides update application, serialization, durability, etc.
- **Partially trusted servers in network core**
  - Trusted to execute protocol, not trusted with user data
Challenges

- Caching, Prefetching, Consistency
- Efficient update encoding, Link scheduling
- Verification, Privacy
Challenges

- Caching, Prefetching, Consistency
- Efficient update encoding, Link scheduling
- Verification, Privacy

![Diagram showing client, cache, and server connections]
Challenges

- Caching, Prefetching, Consistency
- Efficient update encoding, Link scheduling
- Verification, Privacy
Challenges

- Caching, Prefetching, Consistency
- Efficient update encoding, Link scheduling
- Verification, Privacy
Challenges

- Caching, Prefetching, Consistency
- Efficient update encoding, Link scheduling
- Verification, Privacy
Bandwidth-Efficient Peer-to-Peer Storage

- **Goal**
  - Minimize bandwidth to store data
  - Maintain data privacy and verifiability
Outline

• Introduction and Background
• Minimize bandwidth to write data
  – Delta-encoded Updates
  – Efficiency Results
• Maintain privacy and verifiability
  – Data Representation
• Conclusion
Ideal Updates

• Ideally, application provides changes
• Alternatively, extract changes
Common Approaches

• Whole - replace all data on change

• Block - write blocks that have changed
  – Fix-sized blocks
Alternative: Rabin Fingerprints

File Summary
ChunkID, offset, length
1. H(C1), 0, 500
2. H(C2), 500, 500
3. H(C3), 1000, 300
4. H(C4), 1300, 600
5. H(C5), 1900, 700
Computing Changes

\[
\text{Diff} \begin{cases} 
H(C1) & H(C1) \\
H(C2) & H(C6) \\
H(C3) & H(C4) \\
H(C4) & H(C5) \\
H(C5) & \ 
\end{cases}, \quad F' = H(C6)
\]

• Compare list of hashes to find changes
  – Remove C2 and C3, Insert C6
Computing Changes

- Compare list of hashes to find changes
  - Remove C2 and C3, Insert C6
Creating Updates

- Combine list of changes and File Summaries
  - Remove C2 and C3, Insert C6

1. H(C1), 0, 500
2. H(C2), 500, 500
3. H(C3), 1000, 300
4. H(C4), 1300, 600
5. H(C5), 1900, 700

1. H(C1), 0, 500
2. H(C6), 500, 1000
3. H(C4), 1500, 600
4. H(C5), 2100, 700
Creating Updates

- Combine list of changes and File Summaries
  - Remove C2 and C3, Insert C6

1. H(C1), 0, 500
2. H(C2), 500, 500
3. H(C3), 1000, 300
4. H(C4), 1300, 600
5. H(C5), 1900, 700

- Update = (Remove offset=500, length=800), (Insert offset=500, length=1000, E(C6))
Creating Updates

- Combine list of changes and File Summaries
  - Remove C2 and C3, Insert C6
    1. H(C1), 0, 500
    2. H(C2), 500, 500
    3. H(C3), 1000, 300
    4. H(C4), 1300, 600
    5. H(C5), 1900, 700
  
  1. H(C1), 0, 500
  2. H(C6), 500, 1000
  3. H(C4), 1500, 600
  4. H(C5), 2100, 700

- Update = (Remove offset=500, length=800),
  (Insert offset=500, length=1000, E(C6))

- Data = \[ IV \oplus E(C_6) \]
Three Micro-Workloads

• **MS Word**
  – 11 page (700 KB) document
  – Perform global search-and-replace

• **Java development**
  – Source code files from consecutive CVS check-ins
  – 26 files (435 KB) change

• **Email**
  – User receives 5 new messages
  – Mailbox size increases 80 KB -> 100 KB
Overheads

- **Computation of File Summaries**
  - 5 MB of data/second

- **Storage overhead of File Summaries**
  - Word: 4.4 MB of state/GB of user data

- **Update creation**
  - Word: 120 ms
  - 50 ms to encrypt data
Update Size

- Server stores initial version of file
- Measure update size to write new version
• **Measure time to save new version**
  - Compute update, network transmission, execute
  - Results shown for 56 kb network connection
Outline

• Introduction and Background
• Minimize bandwidth to write data
  – Delta-encoded Updates
  – Efficiency Results
• Maintain privacy and verifiability
  – Data Representation
• Conclusion
Data Representation

• Requirements
  – Implement operations used in updates
    • Append, Truncate, Insert, Delete
  – Verifiable
  – Index blocks of variable size
  – Updates cause proportional changes
  – Data treated as opaque
Data Structure

- Build tree over blocks of data
- Reference children by secure hash for verifiability
- Insight: Use *relative* offsets

Root

Leaves

1300
H(L1)
2600
H(L2)

500
H(D1)
1000
H(D2)
1300
H(D3)

600
H(D4)
1300
H(D5)

Data Nodes

E(C1)  E(C2)  E(C3)  E(C4)  E(C5)
Modifying the Data Structure
Conclusion

- P2P storage systems should support weakly-connected clients
- Delta-encoded updates reduce update bandwidth
- Verifiable data structure to execute updates of encrypted data
Related Work

• Low Bandwidth File System (LBFS)
  – Muthitacharoen, Chen, and Mazieres

• CASPER file system
  – Tolia, et. al.

• EXODUS database system
  – Carey, DeWitt, Richardson, and Shekita
Future Work

- Evaluate on larger, real-world workloads

- Study other challenges
  - Efficient sharing between users/devices
  - Write buffering and consistency
  - Support disconnected operation
Improving
Bandwidth Efficiency
of Peer-to-Peer Storage

Patrick Eaton, Emil Ong, and John Kubiatowicz
University of California, Berkeley
http://oceanstore.cs.berkeley.edu/