Antiquity and OpenDHT

Robert Burgess
April 14, 2009

The Real World

Multiple autonomous organizations

Geographically dispersed

All servers eventually fail

Disasters

Churn

The Real World

Multiple autonomous organizations

Geographically dispersed

All servers eventually fail

Disasters

Churn

The Real World

Multiple autonomous organizations

Geographically dispersed

All servers eventually fail

Disasters

Churn

The Real World

Multiple autonomous organizations

Geographically dispersed

All servers eventually fail

Disasters

Churn

The Real World

Multiple autonomous organizations

Geographically dispersed

All servers eventually fail

Disasters

Churn

The Real World

Multiple autonomous organizations

Geographically dispersed

All servers eventually fail

Disasters

Churn

Antiquity

Cooperative storage system

Consistent

Byzantine fault-tolerant

Secure

Highly dynamic

Antiquity

Cooperative storage system

Consistent

Byzantine fault-tolerant

Secure

Highly dynamic

Antiquity

Cooperative storage system

Consistent

Byzantine fault-tolerant

Secure

Highly dynamic

Antiquity

Cooperative storage system

Consistent

Byzantine fault-tolerant

Secure

Highly dynamic

Antiquity

Cooperative storage system

Consistent

Byzantine fault-tolerant

Secure

Highly dynamic

Antiquity

Cooperative storage system

Consistent

Byzantine fault-tolerant

Secure

Highly dynamic

Antiquity

- Part of OceanStore
  - Primary replica service
  - Byzantine agreement
  - Requests ordered and signed
- Implements the storage back-end
- Secure append-only log

Antiquity

- Part of OceanStore
  - Primary replica service
  - Byzantine agreement
  - Requests ordered and signed
- Implements the storage back-end
- Secure append-only log

Antiquity

- Part of OceanStore
  - Primary replica service
  - Byzantine agreement
  - Requests ordered and signed
- Implements the storage back-end
- Secure append-only log

Antiquity

- Part of OceanStore
  - Primary replica service
  - Byzantine agreement
  - Requests ordered and signed
- Implements the storage back-end
- Secure append-only log

Antiquity

- Part of OceanStore
  - Primary replica service
  - Byzantine agreement
  - Requests ordered and signed
- Implements the storage back-end
- Secure append-only log

Antiquity

- Part of OceanStore
  - Primary replica service
  - Byzantine agreement
  - Requests ordered and signed
- Implements the storage back-end
- Secure append-only log

Antiquity

- Part of OceanStore
  - Primary replica service
  - Byzantine agreement
  - Requests ordered and signed
- Implements the storage back-end
- Secure append-only log

OpenDHT

Public Distributed Hash Table service

Also from Berkeley

Goal:

- Provide DHT functionality
- Clients need not be DHT nodes
- Enable very, very simple clients

OpenDHT

Public Distributed Hash Table service

Also from Berkeley

Goal:

- Provide DHT functionality
- Clients need not be DHT nodes
- Enable very, very simple clients

OpenDHT

Public Distributed Hash Table service

Also from Berkeley

Goal:

- Provide DHT functionality
- Clients need not be DHT nodes
- Enable very, very simple clients

OpenDHT

- Public Distributed Hash Table service
- Also from Berkeley
- Goal:
  - Provide DHT functionality
  - Clients need not be DHT nodes
  - Enable very, very simple clients

OpenDHT

Public Distributed Hash Table service

Also from Berkeley

Goal:

- Provide DHT functionality
- Clients need not be DHT nodes
- Enable very, very simple clients

OpenDHT

- Public Distributed Hash Table service
- Also from Berkeley
- Goal:
  - Provide DHT functionality
  - Clients need not be DHT nodes
  - Enable very, very simple clients

OpenDHT

Public Distributed Hash Table service

Also from Berkeley

Goal:

- Provide DHT functionality
- Clients need not be DHT nodes
- Enable very, very simple clients

OpenDHT

OPEN DHT

TURN ON
PUT() IN
GET() OUT

Powered By

PLANETLAB

Comparison

Public storage systems

Wide-area cooperative services

OpenDHT put/get/lookup

Antiquity secure append-only log

OceanStore higher-level object storage

Comparison

Public storage systems

Wide-area cooperative services

OpenDHT put/get/lookup

Antiquity secure append-only log

OceanStore higher-level object storage

Comparison

Public storage systems

Wide-area cooperative services

OpenDHT put/get/lookup

Antiquity secure append-only log

OceanStore higher-level object storage

Comparison

Public storage systems

Wide-area cooperative services

OpenDHT put/get/lookup

Antiquity secure append-only log

OceanStore higher-level object storage

Comparison

Public storage systems

Wide-area cooperative services

OpenDHT put/get/lookup

Antiquity secure append-only log

OceanStore higher-level object storage

Comparison

Public storage systems

Wide-area cooperative services

OpenDHT put/get/lookup

Antiquity secure append-only log

OceanStore higher-level object storage

Antiquity Interface

`create()`

`append()`

`read()`

No explicit deletion; expiration based on explicit (extendable) timestamps

Antiquity Interface

`create()`

`append()`

`read()`

No explicit deletion; expiration based on explicit (extendable) timestamps

Antiquity Interface

`create()`

`append()`

`read()`

No explicit deletion; expiration based on explicit (extendable) timestamps

Antiquity Interface

`create()`

`append()`

`read()`

No explicit deletion; expiration based on explicit (extendable) timestamps

Antiquity Interface

`create()`

`append()`

`read()`

No explicit deletion; expiration based on explicit (extendable) timestamps

Architecture

- Clients
  - Identifed by private keys
  - Assumed fail-stop
- Administrator
  - Assigns storage servers
  - Assumed trusted and non-faulty
- Storage servers
  - Assigns storage servers
  - Up to a threshold may be Byzantine

Architecture

- Clients
  - Identifed by private keys
  - Assumed fail-stop
- Administrator
  - Assigns storage servers
  - Assumed trusted and non-faulty
- Storage servers
  - Assigns storage servers
  - Up to a threshold may be Byzantine

Architecture

- Clients
  - Identifed by private keys
  - Assumed fail-stop
- Administrator
  - Assigns storage servers
  - Assumed trusted and non-faulty
- Storage servers
  - Assigns storage servers
  - Up to a threshold may be Byzantine

Architecture

- Clients
  - Identifed by private keys
  - Assumed fail-stop
- Administrator
  - Assigns storage servers
  - Assumed trusted and non-faulty
- Storage servers
  - Assigns storage servers
  - Up to a threshold may be Byzantine

Architecture

- Clients
  - Identifed by private keys
  - Assumed fail-stop
- Administrator
  - Assigns storage servers
  - Assumed trusted and non-faulty
- Storage servers
  - Assigns storage servers
  - Up to a threshold may be Byzantine

Architecture

- Clients
  - Identifed by private keys
  - Assumed fail-stop
- Administrator
  - Assigns storage servers
  - Assumed trusted and non-faulty
- Storage servers
  - Assigns storage servers
  - Up to a threshold may be Byzantine

Architecture

- Clients
  - Identifed by private keys
  - Assumed fail-stop
- Administrator
  - Assigns storage servers
  - Assumed trusted and non-faulty
- Storage servers
  - Assigns storage servers
  - Up to a threshold may be Byzantine

Architecture

- Clients
  - Identifed by private keys
  - Assumed fail-stop
- Administrator
  - Assigns storage servers
  - Assumed trusted and non-faulty
- Storage servers
  - Assigns storage servers
  - Up to a threshold may be Byzantine

Architecture

- Clients
  - Identifed by private keys
  - Assumed fail-stop
- Administrator
  - Assigns storage servers
  - Assumed trusted and non-faulty
- Storage servers
  - Assigns storage servers
  - Up to a threshold may be Byzantine

Architecture

- Clients
  - Identifed by private keys
  - Assumed fail-stop
- Administrator
  - Assigns storage servers
  - Assumed trusted and non-faulty
- Storage servers
  - Assigns storage servers
  - Up to a threshold may be Byzantine

Design

- Logs
  - Divided into *extents*
    - Immutable
    - Identified by hash of content
    - Bears *verifier* of content and history
      $V_i = H(V_{i-1} + H(D_i))$
  - Special *head extent*
    - Mutable
    - Identifed by hash of owner's public key

Design

Logs

- Divided into *extents*
  - Immutable
  - Identified by hash of content
  - Bears *verifier* of content and history
    $V_i = H(V_{i-1} + H(D_i))$
- Special *head extent*
  - Mutable
  - Identifed by hash of owner's public key

Design

- Logs
  - Divided into *extents*
    - Immutable
    - Identified by hash of content
    - Bears *verifier* of content and history
      $V_i = H(V_{i-1} + H(D_i))$
  - Special *head extent*
    - Mutable
    - Identifed by hash of owner's public key

Design

- Logs
  - Divided into *extents*
    - Immutable
    - Identified by hash of content
    - Bears *verifier* of content and history
      $V_i = H(V_{i-1} + H(D_i))$
  - Special *head extent*
    - Mutable
    - Identifed by hash of owner's public key

Design

- Logs
  - Divided into *extents*
    - Immutable
    - Identified by hash of content
    - Bears *verifier* of content and history
      $$V_i = H(V_{i-1} + H(D_i))$$
  - Special *head extent*
    - Mutable
    - Identifed by hash of owner's public key

Design

- Logs
  - Divided into *extents*
    - Immutable
    - Identified by hash of content
    - Bears *verifier* of content and history
      $V_i = H(V_{i-1} + H(D_i))$
  - Special *head extent*
    - Mutable
    - Identifed by hash of owner's public key

Design

- Logs
  - Divided into *extents*
    - Immutable
    - Identified by hash of content
    - Bears *verifier* of content and history
      $V_i = H(V_{i-1} + H(D_i))$
  - Special *head extent*
    - Mutable
    - Identifed by hash of owner's public key

Design

- Logs
  - Divided into *extents*
    - Immutable
    - Identified by hash of content
    - Bears *verifier* of content and history
      $V_i = H(V_{i-1} + H(D_i))$
  - Special *head extent*
    - Mutable
    - Identifed by hash of owner's public key

Design

- Logs
  - Divided into *extents*
    - Immutable
    - Identified by hash of content
    - Bears *verifier* of content and history
      $V_i = H(V_{i-1} + H(D_i))$
  - Special *head extent*
    - Mutable
    - Identifed by hash of owner's public key

Design

- Two-level naming
  - Extents divided into *blocks*
  - Entries named by tuple $\langle extent, block \rangle$
- Read requires accessing mapping to locate block
- Then `get_blocks()`
- Growth bounded with `snapshot()` and `truncate()`

Design

- Two-level naming
  - Extents divided into *blocks*
  - Entries named by tuple $\langle extent, block \rangle$
- Read requires accessing mapping to locate block
- Then `get_blocks()`
- Growth bounded with `snapshot()` and `truncate()`

Design

- Two-level naming
  - Extents divided into *blocks*
  - Entries named by tuple $\langle extent, block \rangle$
- Read requires accessing mapping to locate block
- Then `get_blocks()`
- Growth bounded with `snapshot()` and `truncate()`

Design

- Two-level naming
  - Extents divided into *blocks*
  - Entries named by tuple $\langle \mathit{extent}, \mathit{block} \rangle$
- Read requires accessing mapping to locate block
- Then `get_blocks()`
- Growth bounded with `snapshot()` and `truncate()`

Design

- Two-level naming
  - Extents divided into *blocks*
  - Entries named by tuple $\langle extent, block \rangle$
- Read requires accessing mapping to locate block
- Then `get_blocks()`
- Growth bounded with `snapshot()` and `truncate()`

Design

- Two-level naming
  - Extents divided into *blocks*
  - Entries named by tuple $\langle \mathit{extent}, \mathit{block} \rangle$
- Read requires accessing mapping to locate block
- Then `get_blocks()`
- Growth bounded with `snapshot()` and `truncate()`

Design

- Two-level naming
  - Extents divided into *blocks*
  - Entries named by tuple $\langle extent, block \rangle$
- Read requires accessing mapping to locate block
- Then `get_blocks()`
- Growth bounded with `snapshot()` and `truncate()`

Quorum Repair

- Restores log replicas to consistent state to fix
  - unavailable quorum due to failures
  - inconsistent quorum due to churn
- Initiated by storage server
  - Server believes repair is needed, notifies administrator
  - Administrator waits for $f + 1$ requests
  - Administrator sends *repair audit* to determine last good state

Quorum Repair

Restores log replicas to consistent state to fix

- unavailable quorum due to failures
- inconsistent quorum due to churn

Initiated by storage server

- Server believes repair is needed, notifies administrator
- Administrator waits for $f+1$ requests
- Administrator sends *repair audit* to determine last good state

Quorum Repair

- Restores log replicas to consistent state to fix
  - unavailable quorum due to failures
  - inconsistent quorum due to churn
- Initiated by storage server
  - Server believes repair is needed, notifies administrator
  - Administrator waits for $f + 1$ requests
  - Administrator sends *repair audit* to determine last good state

Quorum Repair

- Restores log replicas to consistent state to fix
  - unavailable quorum due to failures
  - inconsistent quorum due to churn
- Initiated by storage server
  - Server believes repair is needed, notifies administrator
  - Administrator waits for $f + 1$ requests
  - Administrator sends *repair audit* to determine last good state

Quorum Repair

- Restores log replicas to consistent state to fix
  - unavailable quorum due to failures
  - inconsistent quorum due to churn
- Initiated by storage server
  - Server believes repair is needed, notifies administrator
  - Administrator waits for $f+1$ requests
  - Administrator sends *repair audit* to determine last good state

Quorum Repair

- Restores log replicas to consistent state to fix
  - unavailable quorum due to failures
  - inconsistent quorum due to churn
- Initiated by storage server
  - Server believes repair is needed, notifies administrator
  - Administrator waits for $f + 1$ requests
  - Administrator sends *repair audit* to determine last good state

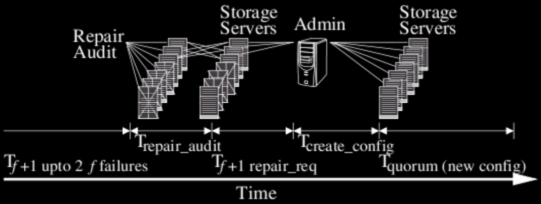
Quorum Repair

- Restores log replicas to consistent state to fix
  - unavailable quorum due to failures
  - inconsistent quorum due to churn
- Initiated by storage server
  - Server believes repair is needed, notifies administrator
  - Administrator waits for $f + 1$ requests
  - Administrator sends *repair audit* to determine last good state

Quorum Repair

- Restores log replicas to consistent state to fix
  - unavailable quorum due to failures
  - inconsistent quorum due to churn
- Initiated by storage server
  - Server believes repair is needed, notifies administrator
  - Administrator waits for $f + 1$ requests
  - Administrator sends *repair audit* to determine last good state

Quorum Repair

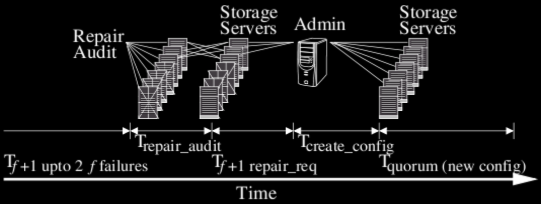


Improved from previous work

Can proceed without full quorum

Consistency nonetheless guaranteed with *soundness proof*

Quorum Repair

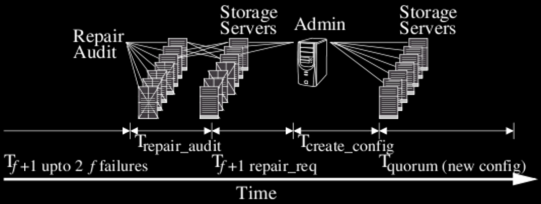


Improved from previous work

Can proceed without full quorum

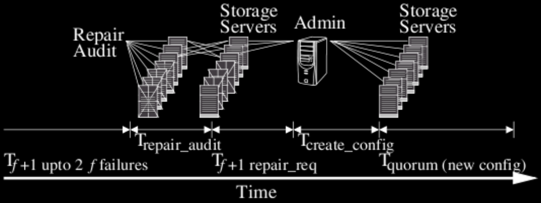
Consistency nonetheless guaranteed with *soundness proof*

Quorum Repair



- Improved from previous work
  - Can proceed without full quorum
  - Consistency nonetheless guaranteed with *soundness proof*

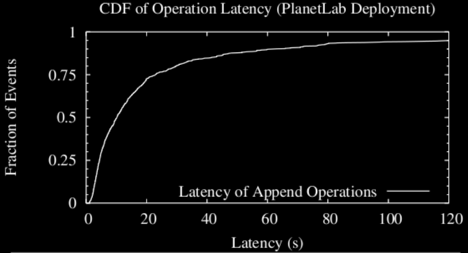
Quorum Repair



Improved from previous work

- Can proceed without full quorum
- Consistency nonetheless guaranteed with *soundness proof*

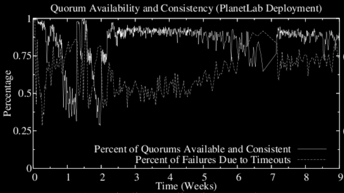
Evaluation



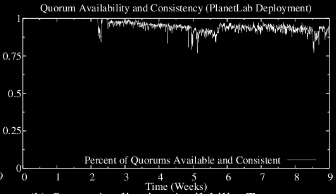
Min	25%	50%	90%	95%	99%	Max
1.08	4.41	10.2	63.1	124.5	302.1	615.9

Figure 11: The latency of operations on PlanetLab varies widely depending on the membership and load of a configuration. As an example, this graphs illustrates the CDF of the latency for appending 32 KB to logs stored in the system. The table highlights key points in the curves.

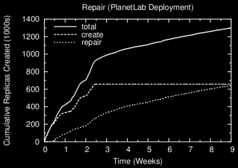
Evaluation



(a) Periodic Application Read



(b) Server Application Availability Trace



Conclusion

Byzantine fault-tolerant agreement

Quorum repair

High availability

High consistency

Secure append-only log
- versioning file system, backups, commitment
- bulletin board in distributed services, etc.

Conclusion

Byzantine fault-tolerant agreement

Quorum repair

High availability

High consistency

Secure append-only log
- versioning file system, backups, commitment
- bulletin board in distributed services, etc.

Conclusion

Byzantine fault-tolerant agreement

Quorum repair

High availability

High consistency

Secure append-only log

- versioning file system, backups, commitment
- bulletin board in distributed services, etc.

Conclusion

Byzantine fault-tolerant agreement

Quorum repair

High availability

High consistency

Secure append-only log
- versioning file system, backups, commitment
- bulletin board in distributed services, etc.

Conclusion

Byzantine fault-tolerant agreement

Quorum repair

High availability

High consistency

Secure append-only log
- versioning file system, backups, commitment
- bulletin board in distributed services, etc.

Conclusion

Byzantine fault-tolerant agreement

Quorum repair

High availability

High consistency

Secure append-only log

versioning file system, backups, commitment

bulletin board in distributed services, etc.

Conclusion

Byzantine fault-tolerant agreement

Quorum repair

High availability

High consistency

Secure append-only log
- versioning file system, backups, commitment
- bulletin board in distributed services, etc.