Ivy: A Read/Write Peer-to-Peer File System

Athicha Muthitacharoen, Robert Morris, Thomer M. Gil, and Benjie Chen

OSDI 2002

Overview

- Motivation
- Introduction
- Challenges
- Design
- Implementation
- Performance
- Discussion

Motivation

- Existing file systems (e.g. NFS, xFS, Harp) exist
  - **Limitation**: Centralization of data or metadata.
  - Their solution: A P2P storage system
- Previous P2P systems (e.g. CFS, PAST)
  - **Limitation**: Mostly read-only or allow only the publisher to modify
  - Their solution: Support multiple writers

Introduction

- Ivy is a multi-user read/write peer-to-peer file system.
- Uses **DHash** for storage.
- Does **NOT** rely on peer storage.
- Does **NOT** trust the users.
- Has **NO** centralized component.

Distributed Hashing

- DHash is a distributed peer-to-peer hash table mapping keys to arbitrary values.

- DHash stores each key/value pair on a set of Internet hosts determined by hashing the key.

Challenges

- Consistency when there are multiple writers
- P2P participants are unreliable. Cannot rely on locking.
- Untrusted participants impose data integrity challenges.
    - Can you *UNDO or IGNORE* undesired modifications?
- It has to deal with network partitions and conflicting updates.

Design

- Ivy uses a set of **logs**.
  - Linked list of immutable records.
- Contains changes to the file system
  - Both metadata and data
- One per participant
  - Records are appended to local log.
  - Records are read from all logs.

Design

- Logs are arranged in **views**
  - A view <u>represents</u> file system state
- Logs are stored in **blocks**
  - Each record is a block.
  - **Log-head** block contains the <u>most recent</u> record
  - View block points to log-heads from each participant

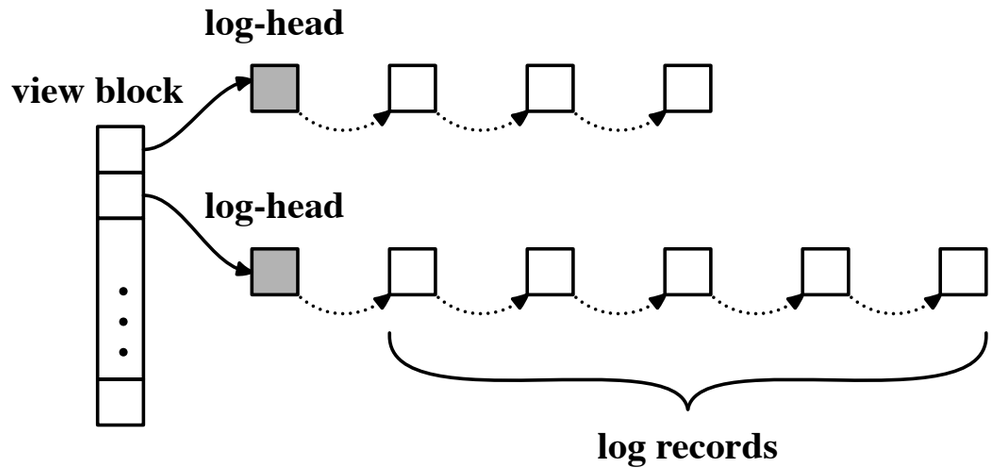


Figure 1: Example Ivy view and logs. White boxes are DHash content-hash blocks; gray boxes are public-key blocks.

Design

- Logs blocks are stored in **DHT**
  - Interface: *put(key, value), get(key)*
  - **Log-head** is a <u>mutable</u> block
    - Identified by the <u>public key</u> of a participant
  - Immutable blocks are content-hashed and cached by participants

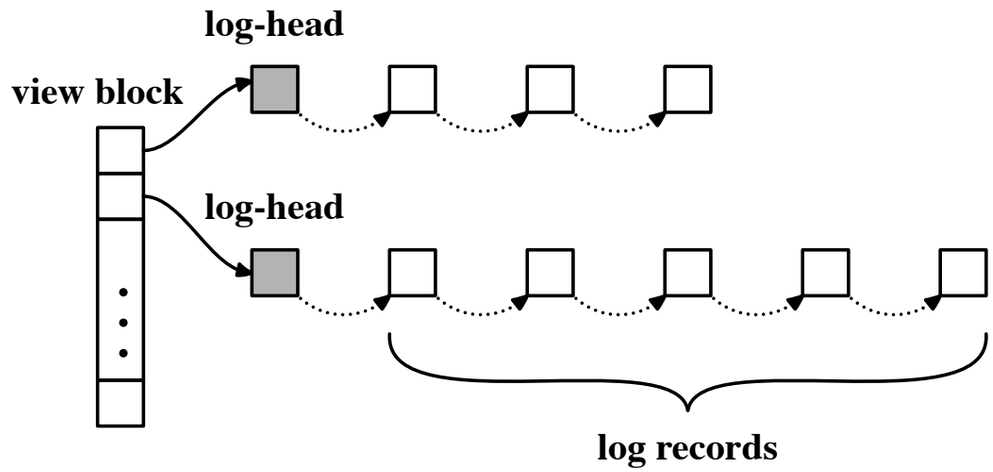


Figure 1: Example Ivy view and logs. White boxes are DHash content-hash blocks; gray boxes are public-key blocks.

Design

- Logs are stored in DHT
  - Per-record replication and authentication
  - Cryptographically signed for verification
    - Each participant stores
      - DHash key of log-head and prev record.
- Ivy uses version vectors for ordering of logs

Field	Use
`prev`	DHash key of next oldest log record
`head`	DHash key of log-head
`seq`	per-log sequence number
`timestamp`	time at which record was created
`version`	version vector

Fields present in all Ivy log records.

Design

- Each log record represents updates from a single file system operation
- Stores 160-bit i-numbers for identifying files and dirs
- Stores attributes and permissions in log record.
    - However, not used to enforce permissions. Relies on encryption.
- Logs are stored indefinitely for security reasons.

Design

Type	Fields	Meaning
`Inode`	type (file, directory, or symlink), i-number, mode, owner	create new inode
`Write`	i-number, offset, data	write data to a file
`Link`	i-number, i-number of directory, name	create a directory entry
`Unlink`	i-number of directory, name	remove a file
`Rename`	i-number of directory, name, i-number of new directory, new file name	rename a file
`Prepare`	i-number of directory, file name	for exclusive operations
`Cancel`	i-number of directory, file name	for exclusive operations
`SetAttrs`	i-number, changed attributes	change file attributes
`End`	none	end of log

Summary of Ivy log record types.

- File system creation (mounts as NFS)
  - Creates Inode record for root dir
- File creation
  - Creates Inode and Link records
- Lookup operation
  - Checks for Unlink record.
- File read operation
  - Checks for Write records. Ignores SetAttrs records.
- File attributes
  - Maintains: mtime, size, ctime, link count
- Dir listing
  - Checks Link records. Ignores Unlink/Rename

Design

- Users create periodical snapshots of the system
  - Incremental and persistent. Stored in DHT
  - Obviates the need for reading entire log.
- Only the participants in the view are trusted.
  - System can recover from malicious modifications using logs.

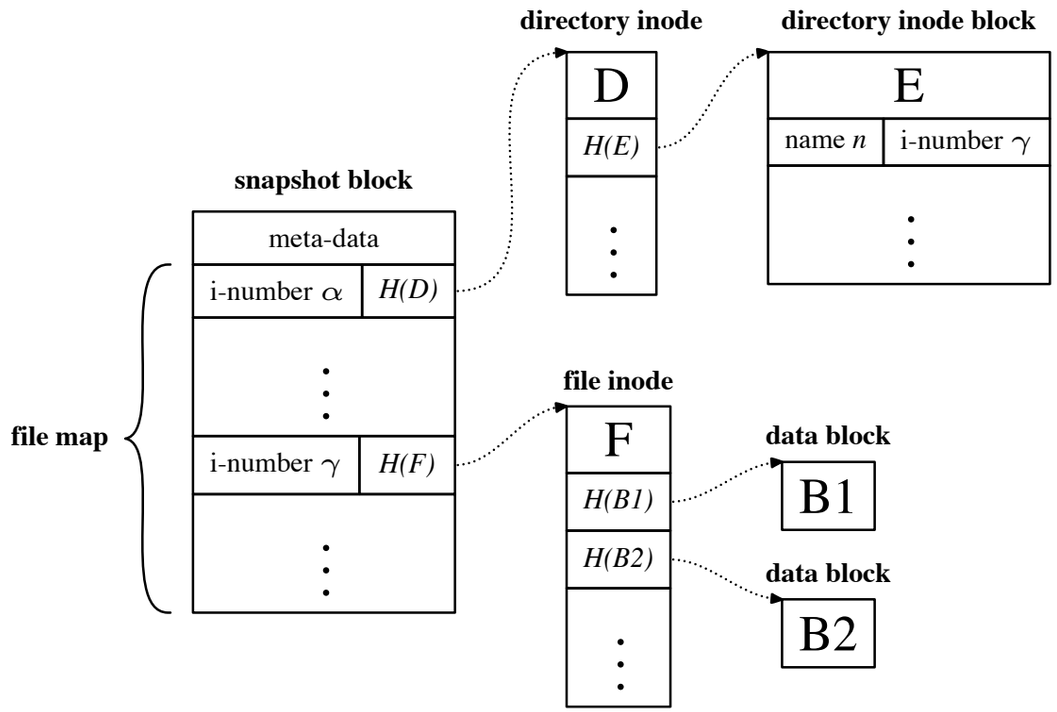


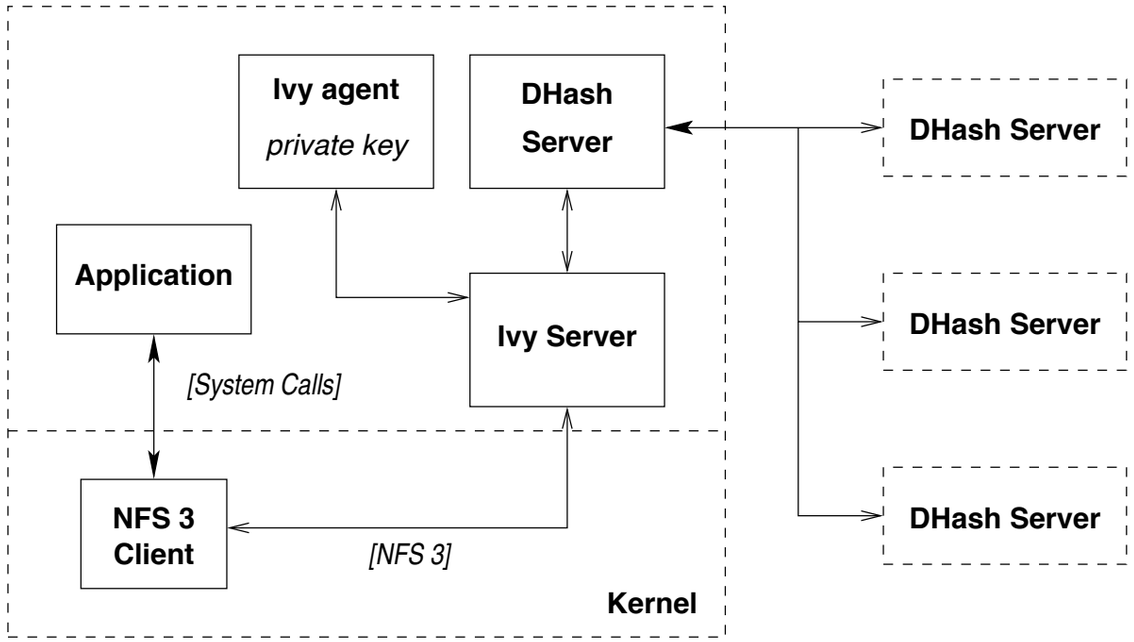
Figure 2: Snapshot data structure. $H(A)$ is the DHash content-hash of A.

Design

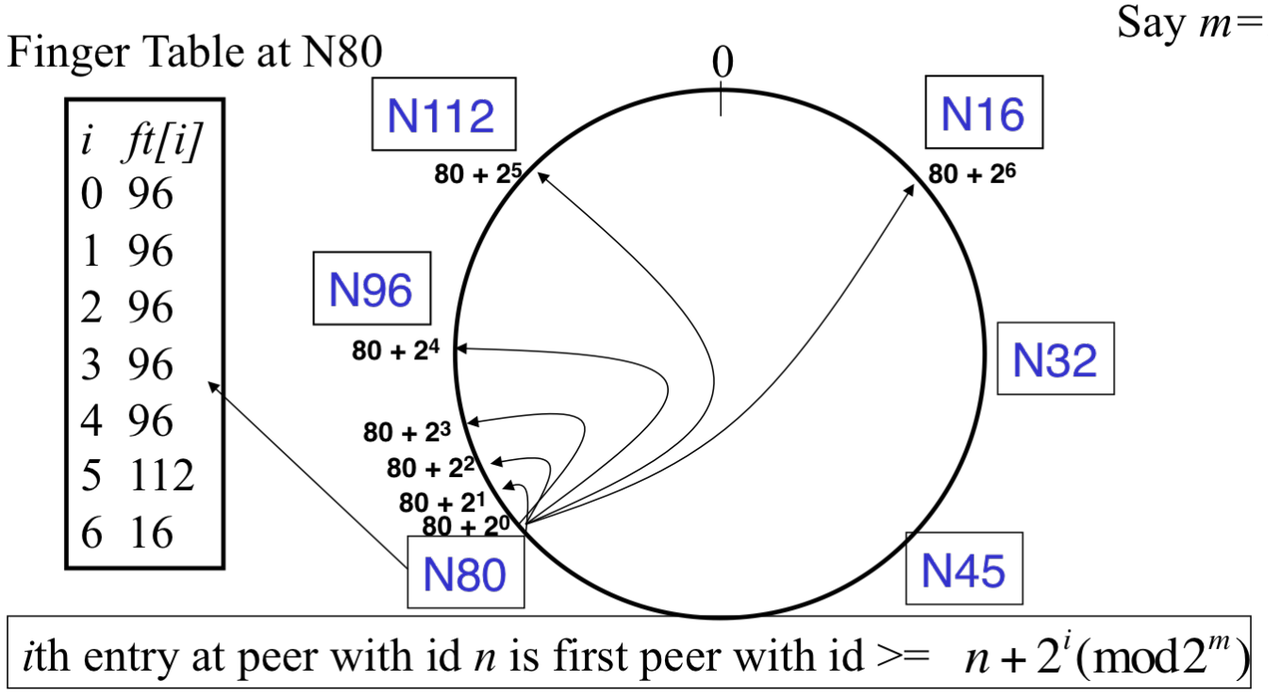
- Provides NFS-like semantics
  - close-to-open consistency
  - writes are deferred until close()
- DHT allows replication
  - In partitioned state, participants can independently modify files. Allows concurrent updates.
    - Exclusive updates done similar to two-phase commit (Prepare record)
  - Uses versioned logs to handle conflicts automatically.
    - Looks for concurrent version vectors.

Architecture

- Ivy appears as a file system (similar to NFS)
  - Acts as loop-back NFS server with in-kernel client.
- Built as a distributed application on top of DHT + Chord.



Chord



Say $m=7$

- Interface: lookup(key) → IP

- O(log N) messages per lookup

Evaluation

- Workload
  - Modified Andrew Benchmark (MAB)
    - create dir tree,
    - copy files into dir tree,
    - walk dir tree and stat each file,
    - read files,
    - compile files

Evaluation

- Setup
  - Block cache size: 512 blocks
  - MAB and Ivy server run on GHz AMD Athlon running FreeBSD 4.5
  - DHash nodes are PlanetLab/RON nodes running FreeBSD 4.5 on 733MHz PIII.
  - No replication (incomplete)

Evaluation

- Configurations
  - Local vs WAN
  - Function of number of:
    - Users,
    - DHash nodes,
    - Concurrent writers,

Evaluation

Phase	**Ivy** (s)	**NFS** (s)
Mkdir	0.6	0.5
Create/Write	6.6	0.8
Stat	0.6	0.2
Read	1.0	0.8
Compile	10.0	5.3
Total	18.8	7.6

Table 3: Real-time in seconds to run the MAB with a single Ivy log and all software running on a single machine. The NFS column shows MAB run-time for NFS over a LAN.

Phase	**Ivy** (s)	**NFS** (s)
Mkdir	11.2	4.8
Create/Write	89.2	42.0
Stat	65.6	47.8
Read	65.8	55.6
Compile	144.2	130.2
Total	376.0	280.4

Table 4: MAB run-time with four DHash servers on a WAN. The file system contains four logs.

Single User MAB on LAN

386 NFS RPCs
508 log updates

Ivy inserts 8.8MB for 1.6MB data
SHA1 expensive

Single User MAB on WAN

(DHT RTT 9, 16, 82 ms)

1 NFS req causes 3 log-head fetches
(total fetches: 3346, bounded: 82ms)

Ivy slower due to more n/w trips

Evaluation

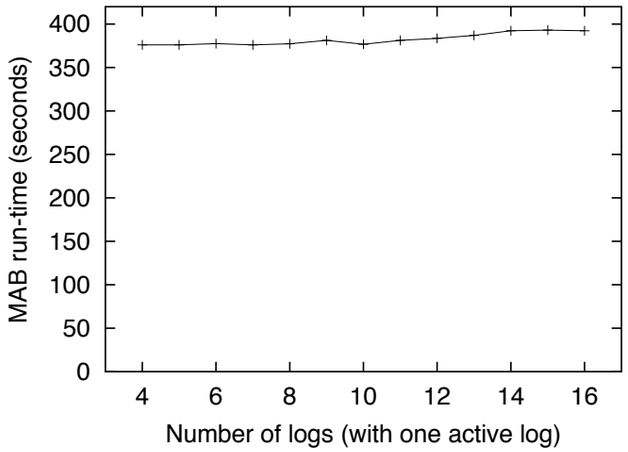


Figure 5: MAB run-time as a function of the number of logs. Only one participant is active.

Many logs, one writer

Little impact: logs are fetched in parallel

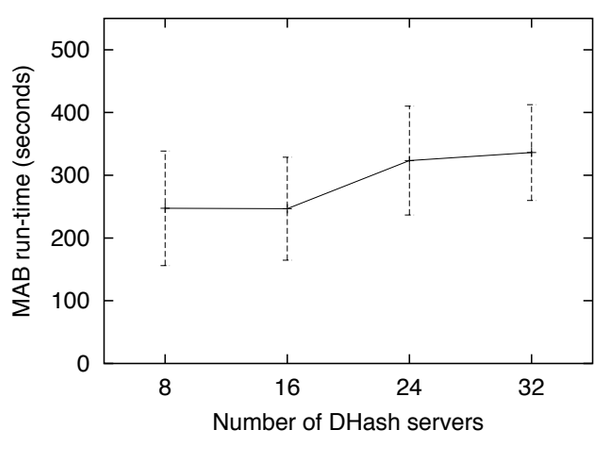


Figure 6: Average MAB run-time as the number of DHash servers increases. The error bars indicate standard deviation over different choices of PlanetLab hosts and different mappings of blocks to DHash servers.

Many DHash Servers (host to servers avg rtt: 32ms)

Runtime grows due to rtf to servers

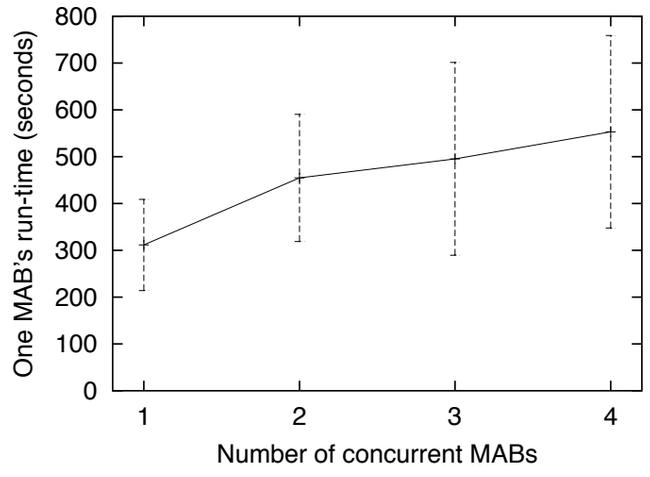


Figure 7: Average run-time of MAB when several MABs are running concurrently on different hosts on the Internet. The error bars indicate standard deviation over all the MAB runs.

Many Writers

Runtime grows because participants fetch each other's log heads

Discussion

- Ivy does not **reclaim log storage space**
  - Ivy relies on logs to make updates. Discarding logs to reclaim space can hurt data security.
  - With storage getting cheaper now, this design decision may not turn out to be too expensive

Discussion

- Ivy provides <u>automatic</u>, application-specific conflict resolution when partition heals.
- Uses application tools for resolution
- This may not work for all applications.

Discussion

- 160-bit i-numbers are generated randomly for files independently at each participant to minimize the probability of collision.

- What if the same i-numbers are allocated for different files.