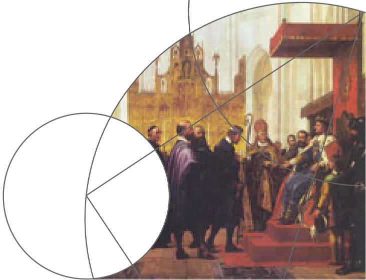
Faculty of Science

Hindsight: A Prototype for a Flexible, Secure and Crash-safe Backup System

Johan Brinch   Morten Brøns-Pedersen
Institute of Computer Science, University of Copenhagen



March 9, 2012

Program

Program

- Background

Program

- Background
- Goals:
    - Flexible (front-end, back-end)
    - Efficient (time usage, space usage)
    - Secure (privacy, authentication)

Program

- Background
- Goals:
  - Flexible (front-end, back-end)
  - Efficient (time usage, space usage)
  - Secure (privacy, authentication)
- System design:
  - Insert
  - Retrieve
  - Delete

Program

- Background
- Goals:
  - Flexible (front-end, back-end)
  - Efficient (time usage, space usage)
  - Secure (privacy, authentication)
- System design:
  - Insert
  - Retrieve
  - Delete
- Demo!

Program

- Background
- Goals:
    - Flexible (front-end, back-end)
    - Efficient (time usage, space usage)
    - Secure (privacy, authentication)
- System design:
    - Insert
    - Retrieve
    - Delete
- Demo!
- Questions?

Background

Who are we, and what is this?

Background

Who are we, and what is this?

- Students at DIKU

Background

Who are we, and what is this?

- Students at DIKU
- Master thesis project

Background

Who are we, and what is this?

- Students at DIKU
- Master thesis project
- Not just proof of concept

Background

Who are we, and what is this?

- Students at DIKU
- Master thesis project
- Not just proof of concept
- Release candidate: August

Background

Who are we, and what is this?

- Students at DIKU
- Master thesis project
- Not just proof of concept
- Release candidate: August (we hope)

Goals

Scope

Goals

Scope

- Personal computer

Goals

Scope

- Personal computer
- No sharing between systems

Goals

Scope

- Personal computer
- No sharing between systems
- Remote storage, over the Internet

Goals

Scope

- Personal computer
- No sharing between systems
- Remote storage, over the Internet
- Hostile environment

Goals

Scope

- Personal computer
- No sharing between systems
- Remote storage, over the Internet
- Hostile environment/user

Goals

Scope

- Personal computer
- No sharing between systems
- Remote storage, over the Internet
- Hostile environment/user
- (At least) daily backup

Goals

A bunch of systems already, why a new one?

Goals

A bunch of systems already, why a new one?

- Open source

Goals

A bunch of systems already, why a new one?

- Open source
- Flexible back-end

Goals

A bunch of systems already, why a new one?

- Open source
- Flexible back-end
- Global de-duplicity

Goals

A bunch of systems already, why a new one?

- Open source
- Flexible back-end
- Global de-duplicity
- Scalability
    - Total log data
    - Per-snapshot overhead
    - Search and checkout

Goals

A bunch of systems already, why a new one?

- Open source
- Flexible back-end
- Global de-duplicity
- Scalability
  - Total log data
  - Per-snapshot overhead
  - Search and checkout
- Deletion

Goals

A bunch of systems already, why a new one?

- Open source
- Flexible back-end
- Global de-duplicity
- Scalability
  - Total log data
  - Per-snapshot overhead
  - Search and checkout
- Deletion
- Crash-safe

System design

Overview

System design

Overview

- A simple front-end API (key, metadata, value);

System design

Overview

- A simple front-end API (key, metadata, value);
- Idempotent key insertion and crash safety;

System design

Overview

- A simple front-end API (key, metadata, value);
- Idempotent key insertion and crash safety;
- Global de-duplicity (content-based addressing);

System design

Overview

- A simple front-end API (key, metadata, value);
- Idempotent key insertion and crash safety;
- Global de-duplicity (content-based addressing);
- Concurrent b-trees everywhere. No transactions;

System design

Overview

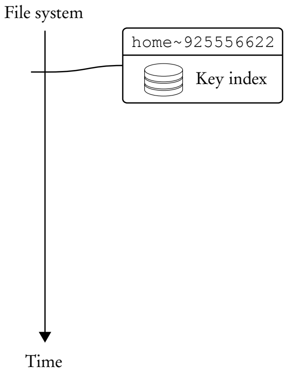
- A simple front-end API (key, metadata, value);
- Idempotent key insertion and crash safety;
- Global de-duplicity (content-based addressing);
- Concurrent b-trees everywhere. No transactions;
  - $\rightarrow$ QuickCheck'ed. . . of course.

System design

Hindsight system overview...

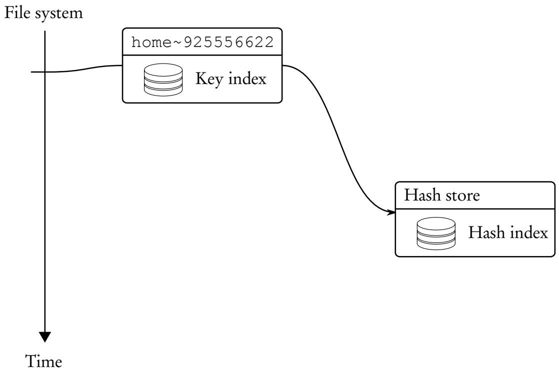
System design

Overview



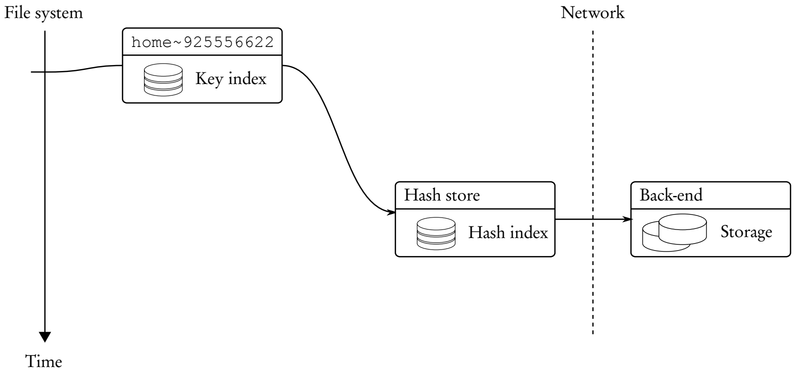
System design

Overview



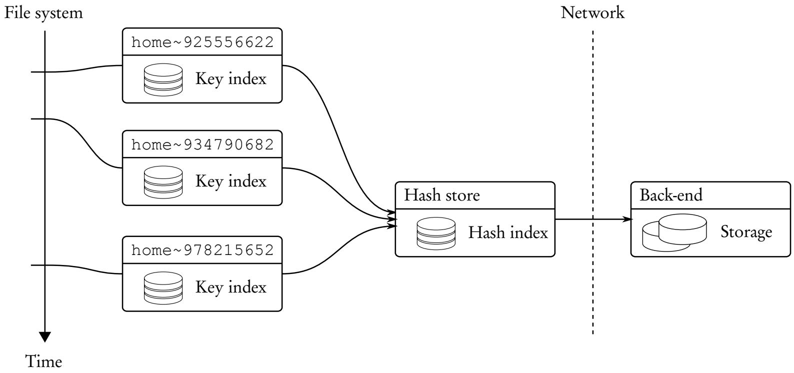
System design

Overview



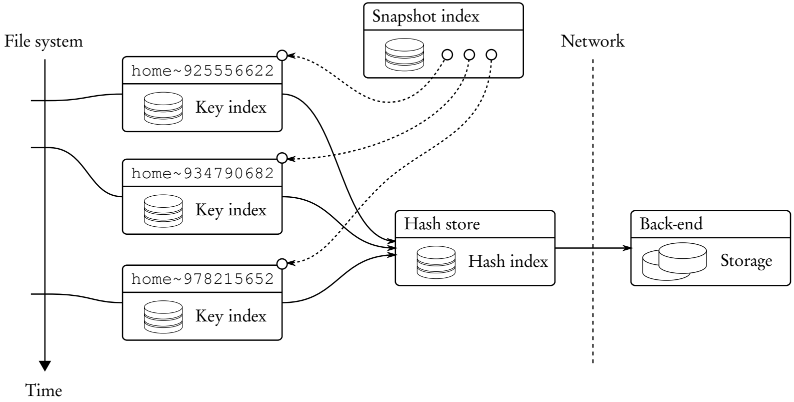
System design

Overview



System design

Overview

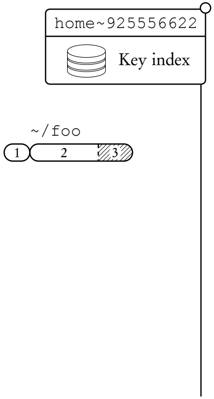


System design

How we store files...

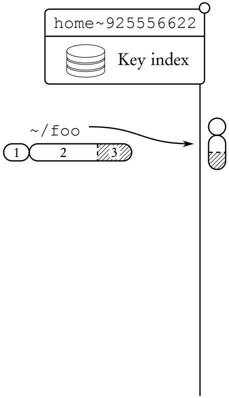
System design

Insertion



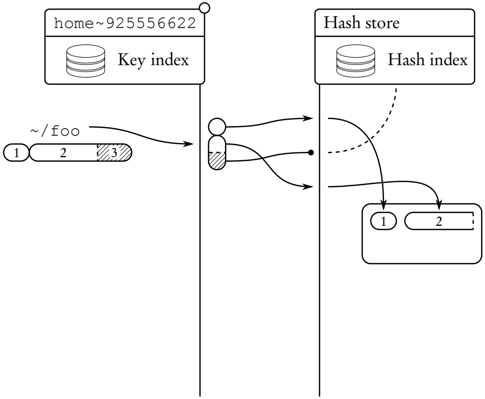
System design

Insertion



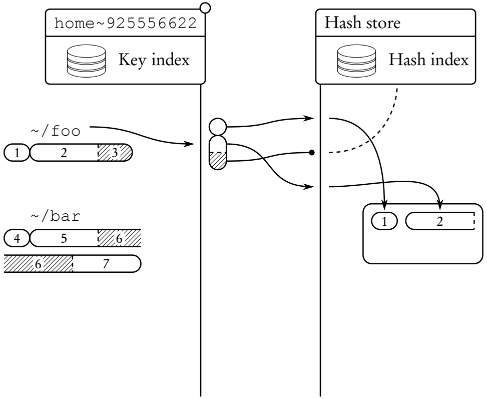
System design

Insertion



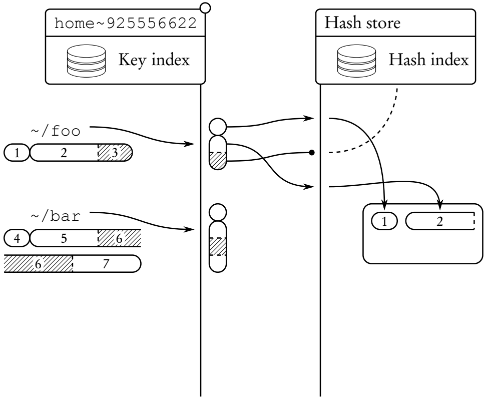
System design

Insertion



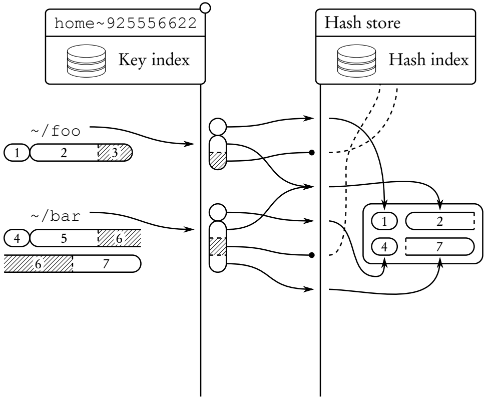
System design

Insertion



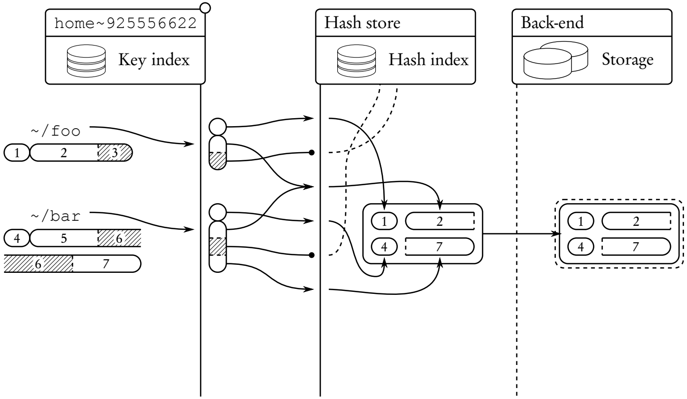
System design

Insertion



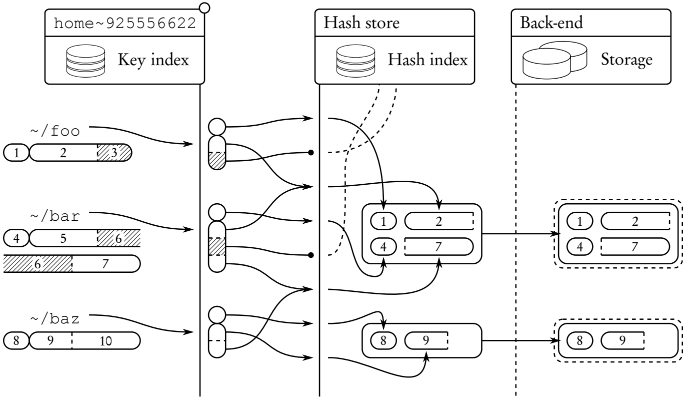
System design

Insertion



System design

Insertion



System design

Log data

- Now we just have the log data to take care of.

System design

Log data

- Now we just have the log data to take care of.
  - → Much smaller than the data we backed up;

System design

Log data

- Now we just have the log data to take care of.
  - $\rightarrow$ Much smaller than the data we backed up;
  - $\rightarrow$ Multiple files with redundant data across snapshots;

System design

Log data

- Now we just have the log data to take care of.
  - → Much smaller than the data we backed up;
  - → Multiple files with redundant data across snapshots;
- Example: 10,278 files, 14 GB
  - → key index: 64 files, 1.2 MB
  - → hash index: 157 files, 1.3 MB

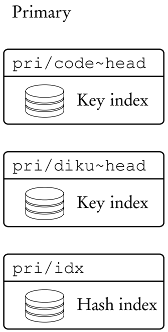
System design

I wish we had some system for backing up files. . .

System design

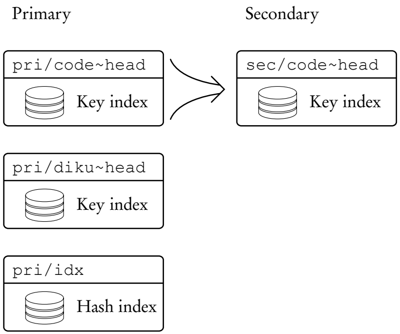
System design

Log data



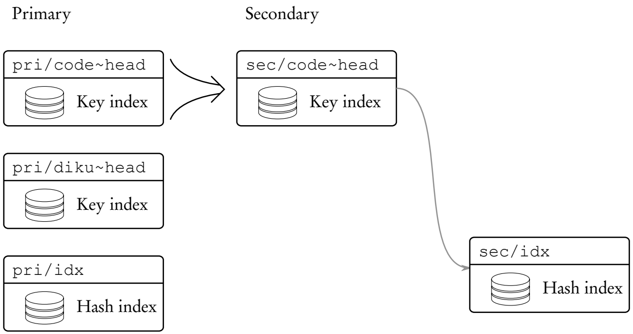
System design

Log data



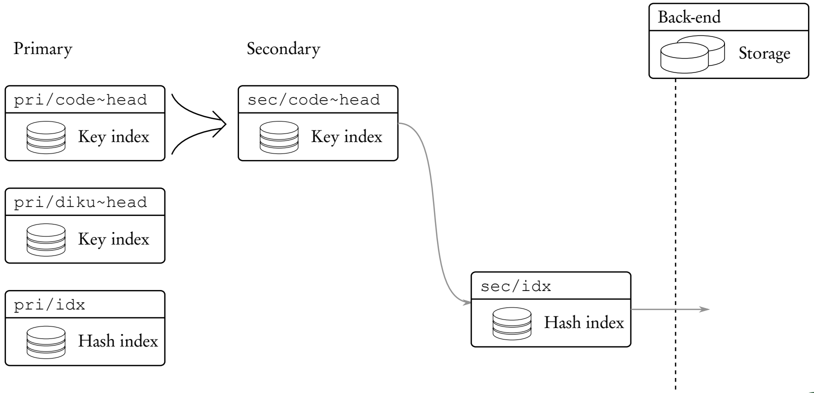
System design

Log data



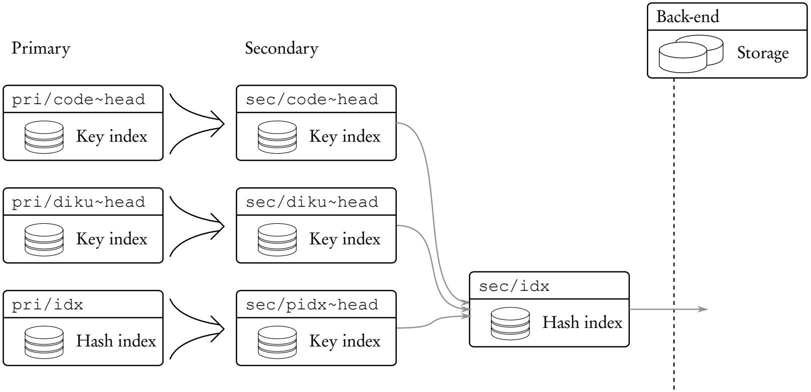
System design

Log data



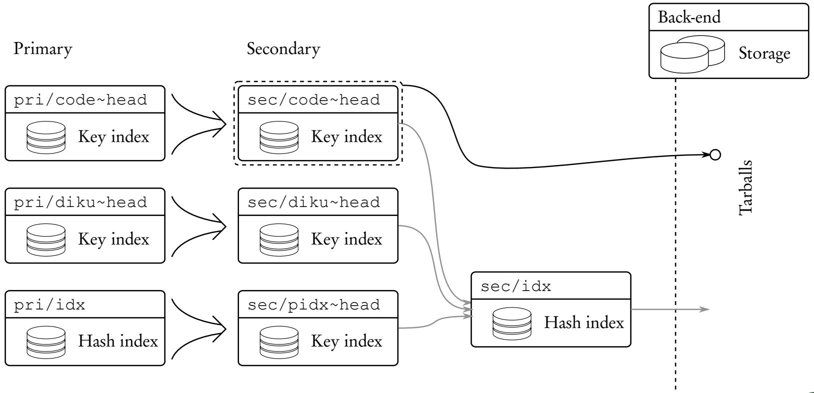
System design

Log data



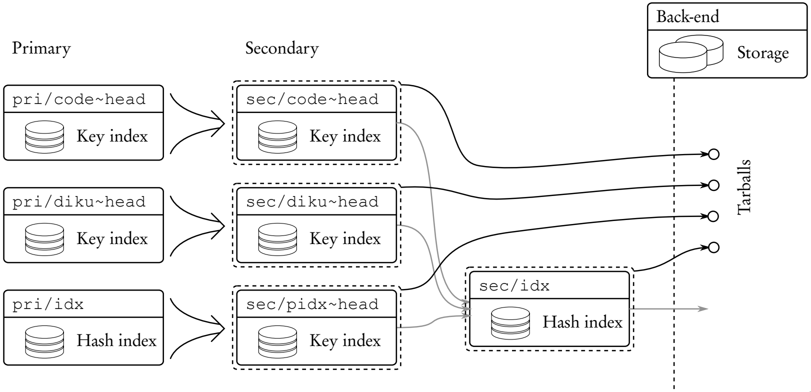
System design

Log data



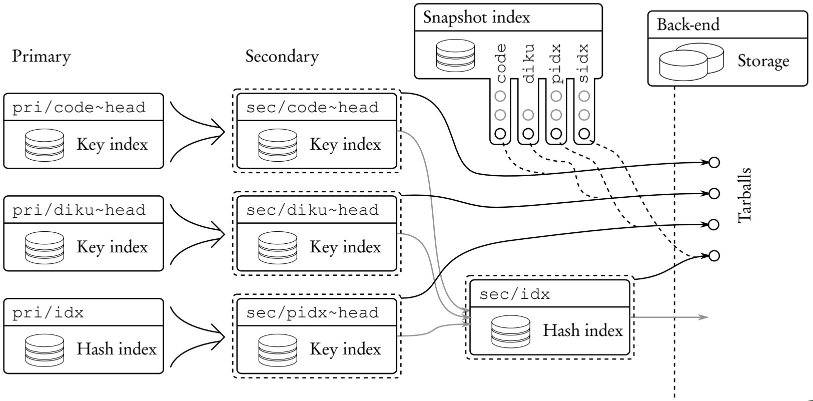
System design

Log data



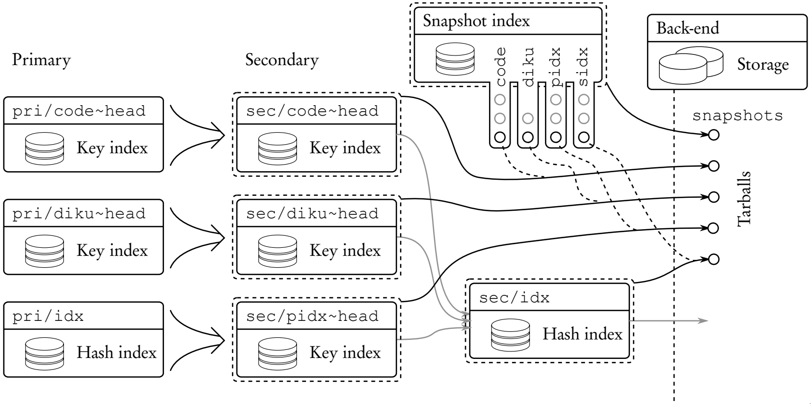
System design

Log data



System design

Log data



System design

Search and retrieve

- B-trees support lazy retrieval of nodes.

System design

Search and retrieve

- B-trees support lazy retrieval of nodes.
- Example using Amazon S3:

System design

Search and retrieve

- B-trees support lazy retrieval of nodes.
- Example using Amazon S3:

```
jos@laptux ~ > time hindsight list diku | wc -l
104391
real   26m00.28s
```

System design

Search and retrieve

- B-trees support lazy retrieval of nodes.
- Example using Amazon S3:

```
jos@laptux ~ > time hindsight list diku | wc -l
104391
real   26m00.28s

jos@laptux ~ > time hindsight list diku:speciale | wc -l
14597
real    3m20.21s
```

System design

Search and retrieve

- B-trees support lazy retrieval of nodes.
- Example using Amazon S3:

```
jos@laptux ~ > time hindsight list diku | wc -l
104391
real   26m00.28s

jos@laptux ~ > time hindsight list diku:speciale | wc -l
14597
real    3m20.21s

jos@laptux ~ > time hindsight listdir diku:speciale | wc -l
10
real    0m19.56s
```

System design

Deletion

- We want to delete unused file blocks.

System design

Deletion

- We want to delete unused file blocks.
  - $\rightarrow$ No transactions; no reference counting.

System design

Deletion

- We want to delete unused file blocks.
  - $\rightarrow$ No transactions; no reference counting.
  - $\rightarrow$ But that is what the others are doing :/

System design

Deletion

- We want to delete unused file blocks.
  - → No transactions; no reference counting.
  - → But that is what the others are doing :/
- Reference sets! (aka reference listing)

System design

Deletion

- We want to delete unused file blocks.
  - $\rightarrow$ No transactions; no reference counting.
  - $\rightarrow$ But that is what the others are doing :/
- Reference sets! (aka reference listing)
  - $\rightarrow$ Keep track of all references, not just the count.

System design

Deletion

- We want to delete unused file blocks.
  - $\rightarrow$ No transactions; no reference counting.
  - $\rightarrow$ But that is what the others are doing :/
- Reference sets! (aka reference listing)
  - $\rightarrow$ Keep track of all references, not just the count.
  - $\rightarrow$ Idempotent operations (insert / remove).

System design

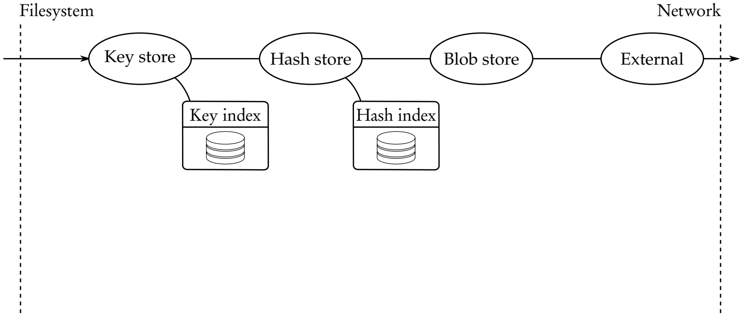
Deletion

- We want to delete unused file blocks.
  - → No transactions; no reference counting.
  - → But that is what the others are doing :/
- Reference sets! (aka reference listing)
  - → Keep track of all references, not just the count.
  - → Idempotent operations (insert / remove).
  - → Tests say the space overhead is acceptable.

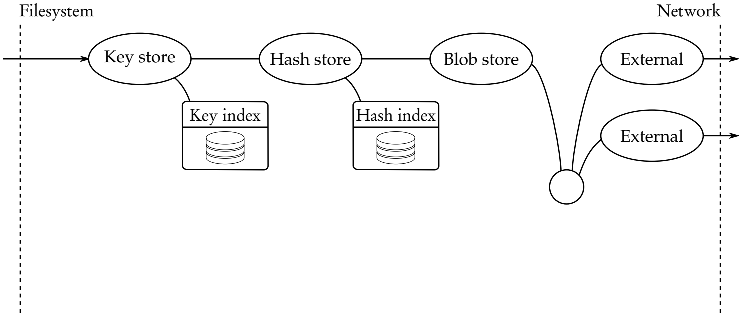
System design

Implementation:
Haskell alá Erlang: Processes and message passing.

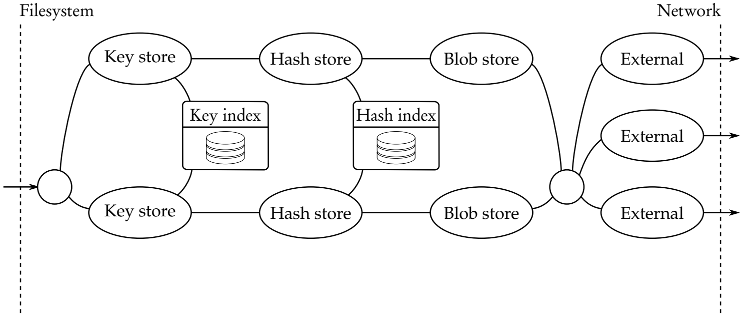
System design



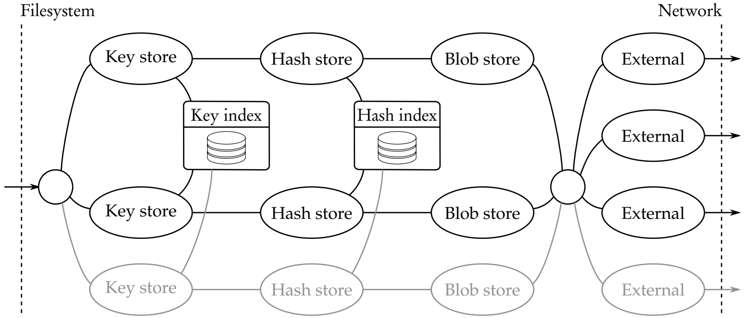
System design



System design



System design



Demo!

Initiate kill-test now!

Questions?