Keso

Keso, A scalable, reliable and secure read/write peer-to-peer file system

Mattias Amnefelt
Johanna Svenningsson

Keso

Master Thesis at IMIT, KTH

- Examiner: Thomas Sjöland
- Supervisor: Luc Onana Alima
- Opponent: Joanna Kühn

Keso

Master Thesis at IMIT, KTH

- Goal:
  - Design a read/write file system suited for real world usage.
- The project:
  - Literature study
  - Design of Keso
  - Implementation of DKS
  - Partial implementation of Keso

Keso

This presentation

- Background
- DKS
- Keso

Keso

What is Keso?

- Keso is a distributed file system built on a peer-to-peer infrastructure.
  - Completely decentralized
  - Scalable
  - Secure
  - Self-organizing
  - Designed for real-world usage

Keso

Why peer-to-peer?

- Fault tolerant
- Scalable
- Makes use of unused resources

Keso

Unused resources

Measurements taken at the IT-department (IT-Enheten) at KTH.

Results:

- 50% of local hard drives unused on workstations
- 3.5 times as much free disk on workstations as was stored in the their distributed file system
- 24% of the data on the file servers was redundant

Keso

How Keso works

- Runs on workstations
- Files split into blocks and distributed over the participating nodes
- Uses a combination of symmetric and asymmetric encryption
- Data blocks and directories are replicated to *f* nodes to provide redundancy
- Built on top of the DKS overlay network

Keso

Implementation

- Made in C++
- Supports all basic file system operations – read, write, delete, mkdir, rmdir...
- No access control
- No kernel support

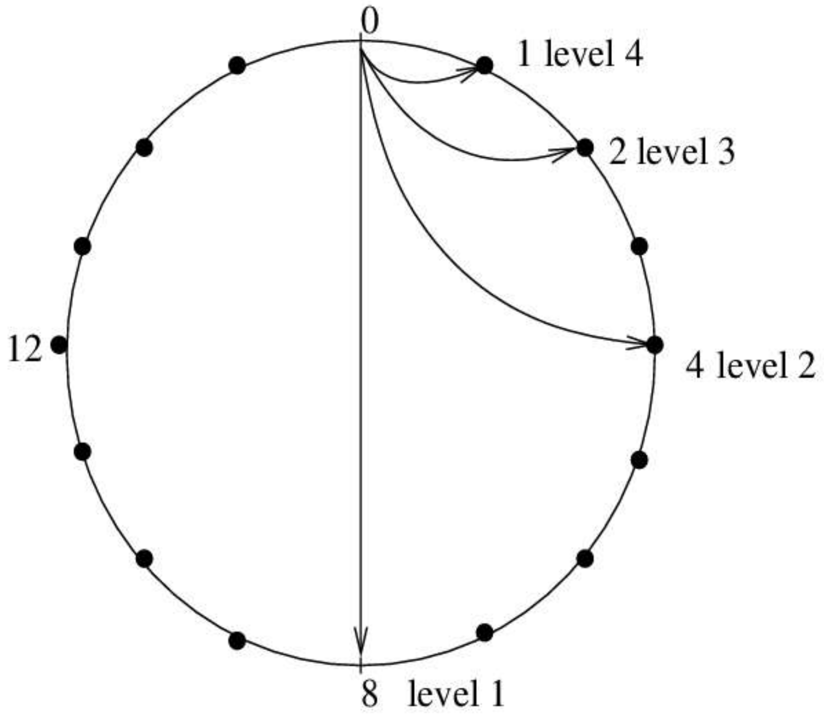
The DKS overlay network

Overview of DKS

- Logical network on top of the underlying network
- Distributed Hash Table
- Small routing tables
- Self-organizing
- Strong guarantees
- Built-in replication

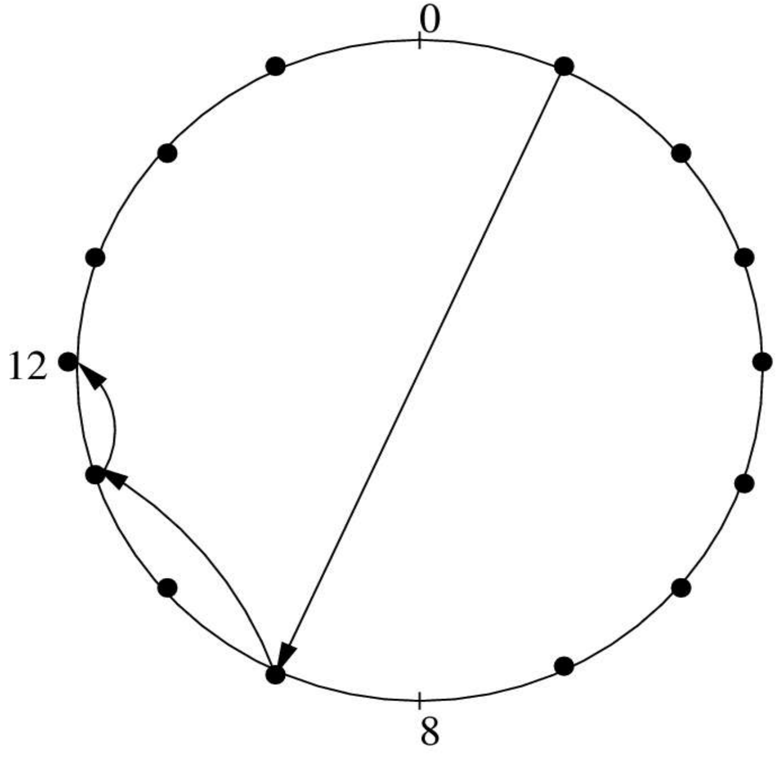
The DKS overlay network

- Nodes are assigned identifiers
- Organized in a ring
- Pointers are kept to nodes at exponentially increasing distance



The DKS overlay network

- Data is assigned keys from the same identifier space
- Stored at the node with the closest succeeding identifier
- With each hop, the distance to the destination is at least decreased by half



Keso

Design objectives

- Keso should
  - Make use of unutilized resources
  - Avoid storing redundant data
  - Scale well and support thousands of clients
  - Be self-organizing
  - Be a secure file system suited for a real-world environment

Keso

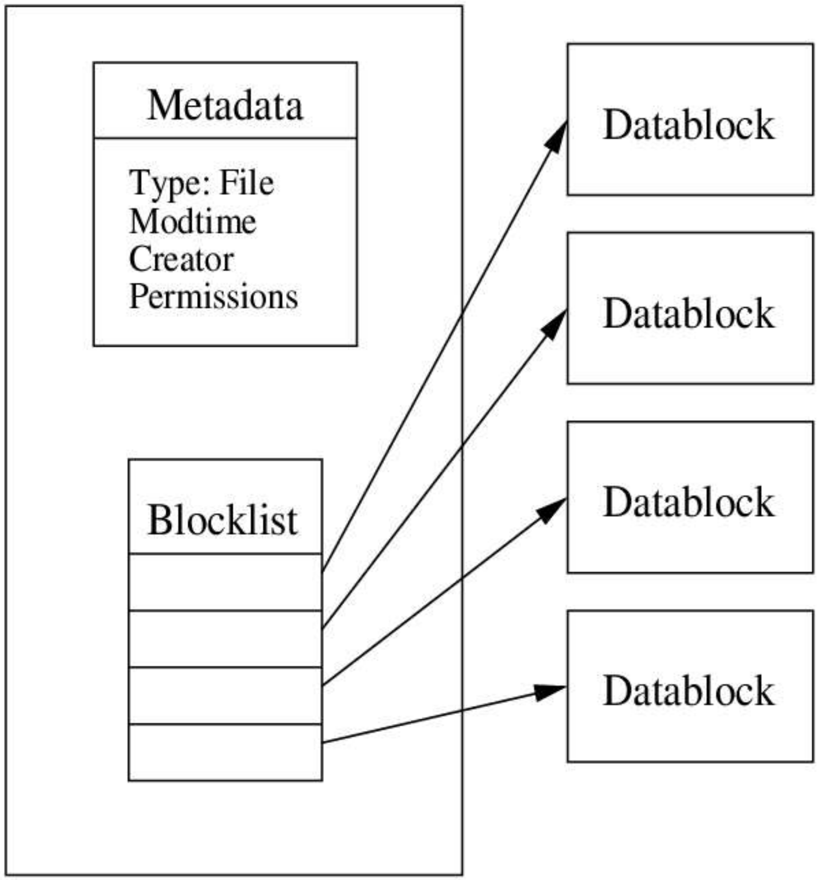
Overview

- Directories and files
- Static and content hash keys
- Old versions of files kept in the file system
- Data is encrypted in a way that avoid storing unnecessarily redundant data

Keso

Overview of files

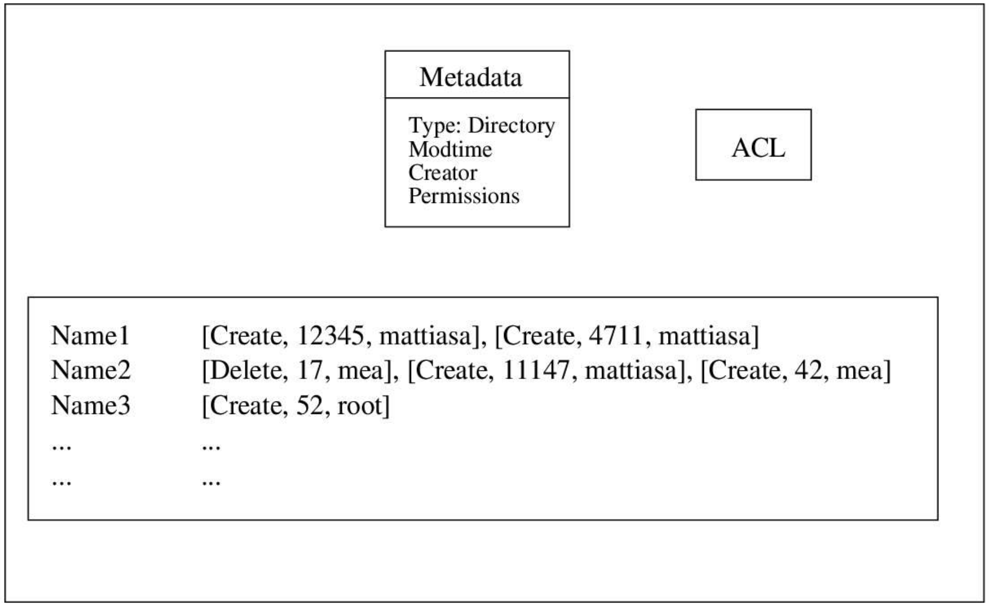
- Data is split into blocks of equal size
- Blocks are referenced from a block list in the inode
- Both blocks and inodes are stored in DKS using a hash of their contents
- All files which contain the same data reference the same blocks



Keso

Overview directories

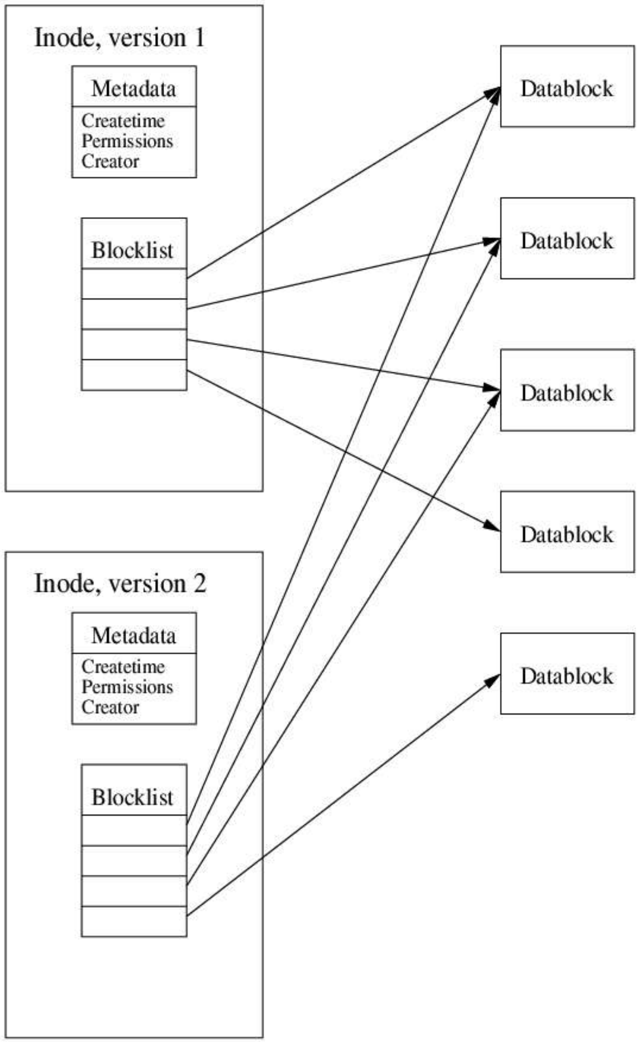
- Acts as a name/inode lookup service
- Identifiers never changes



Keso

Versioning

- All versions of files are kept
- Users can go back through a file's history
- Directories contain a list of file versions
- Only blocks which are changed must be stored additionally



Security in Keso

- Access control
- Data privacy
- Tamper protection

Security in Keso

Access control

- PKI – each user and node has a public/private key pair
- Each directory has a symmetric key used for protecting data in that directory
- The symmetric key for a directory is encrypted with the public keys of all users and groups permitted to access files in that directory

Security in Keso

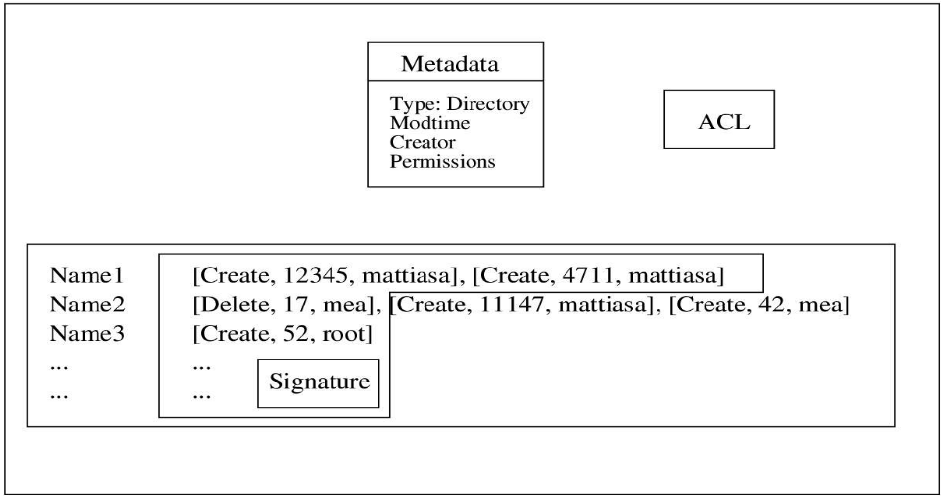
Data privacy

- Each file is encrypted using its own content hash
- The encrypted block is stored in DKS using the content hash of the cipher text
- Both the hash of the clear text and cipher text blocks are stored in the inode
- The inode is finally encrypted with the symmetric directory key.

Security in Keso

Tamper protection

- Data blocks and inodes are stored using the hashes of their contents
- When changes are committed to the directory, the entire latest version and the change is signed
- This makes sure that changes can be tracked through time.



Keso

Storing data

- Data is replicated on a number of nodes using the replication scheme of DKS
- When nodes store data they send acknowledgments to the ”client” node. The ”client” node waits until enough nodes have acknowledged that they have saved the data.

Keso

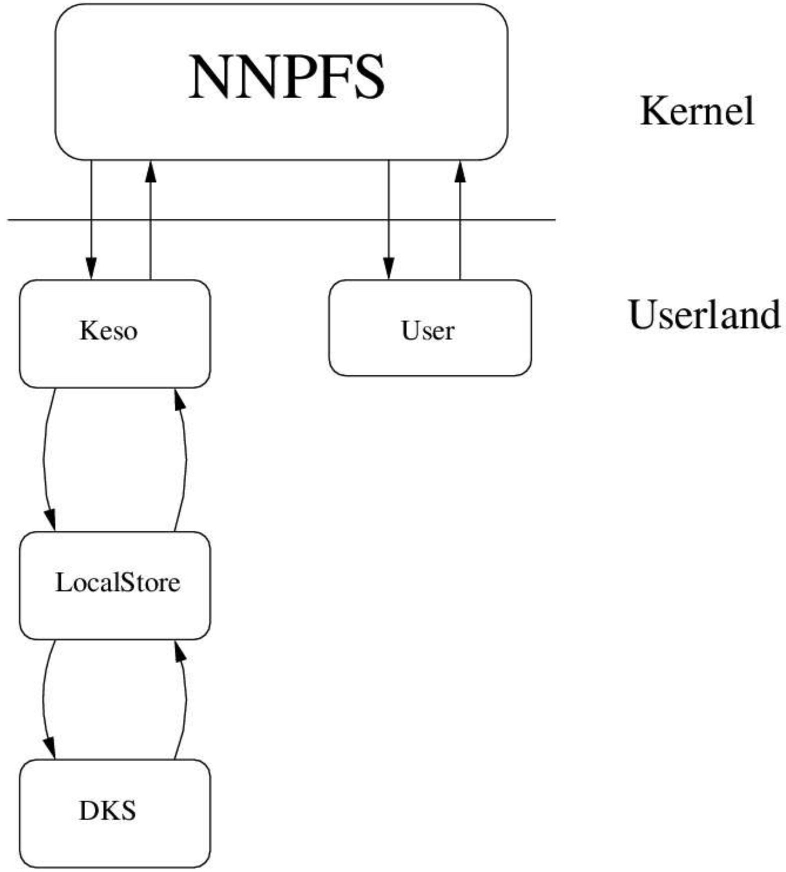
Implementation

- Three separate modules
  - DKS

    Communication
  - LocalStore

    Storing data
  - Keso

    Knowledge about the file system structure



Conclusion

Main achievements

- Design and implementation of a decentralized, scalable and fault-tolerant read/write file system on top of an overlay network such as DKS.
- Provide access control, data privacy and tamper protection while avoiding unnecessarily storing redundant data.
- Collected statistics which show that our design is reasonable in the real world.

Conclusion

Future work

- Complete implementation
- Kernel interaction
- Quota
- Conflict resolution

Conclusion

Questions?

Security in Keso

Data privacy