libchop, a library for distributed storage & data backup

Ludovic Courtès *<ludo@gnu.org>*

GNU Hackers Meeting, Den Haag, 24 July 2010

- **The Basics**
  - inception & rationale
  - storing to remote, untrusted sites
  - library & tools for data storage
- The Mechanics
- The Plan

inception & rationale

- PhD in 2004–2007, LAAS-CNRS, France
- cooperative backup for the people!
- flexible & self-managed distributed storage
- storage to remote (untrusted) sites

storing to remote, untrusted sites

- dissemination of data fragments
- guarantees for data integrity, confidentiality, & authenticity

library & tools for data storage

- content-addressable storage ("deduplication")
- encryption via content-hash keys
- file chopping
- compression
- ...

from the command line

```
$ chop-archiver --archive holiday-pic.jpg
tree_indexer:hash_block_fetcher:hash_index_handle:64:sha1:onx5ee…

…

$ chop-archiver --restore \
  tree_indexer:hash_block_fetcher:hash_index_handle:64:sha1:onx5ee… \
  > restored-pic.jpg
```

from C

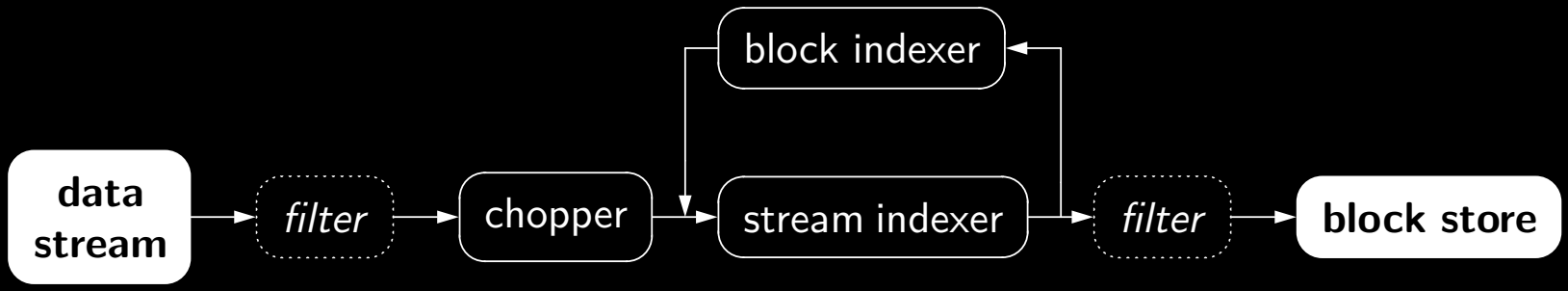
```
chop_indexer_t *indexer;
chop_chopper_t *chopper;
chop_block_indexer_t *block_indexer;
chop_block_store_t *store;
chop_index_handle_t *index;

/* Instantiate these things… */

err = chop_indexer_index_blocks (indexer, chopper,
                                 block_indexer,
                                 store, store,
                                 index);
```

from Guile Scheme

```
(define (archive-file file store)
  ;; Chop FILE and write the blocks to STORE.
  ;; Return an ASCII string denoting its index.
  (let* ((f  (file-stream-open file))
         (c  (anchor-based-chopper-open f))
         (bi (hash-block-indexer-open
                'hash-method/sha1))
         (i  (tree-indexer-open 100)))
    (let ((index (indexer-index-blocks i c bi
                                       store store)))
      (index-handle-ascii-serialize index))))
```

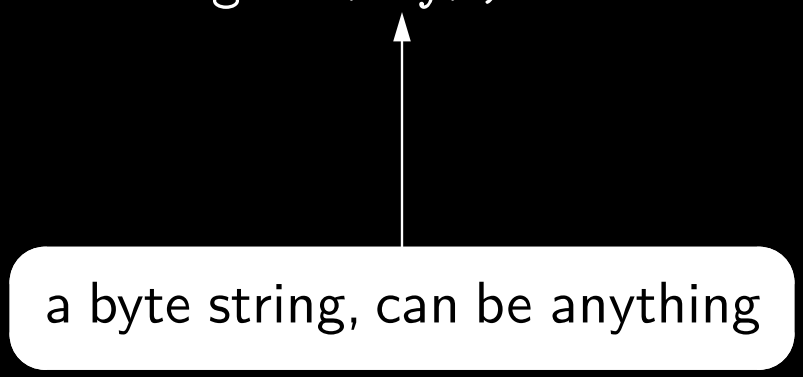
- The Basics
- **The Mechanics**
  - the storage pipeline
  - block stores
  - fixed-size chopper
  - content-based chopper
  - filters
  - tradeoffs
- The Plan

the storage pipeline



block stores

```
put (key, data);

get (key);
```

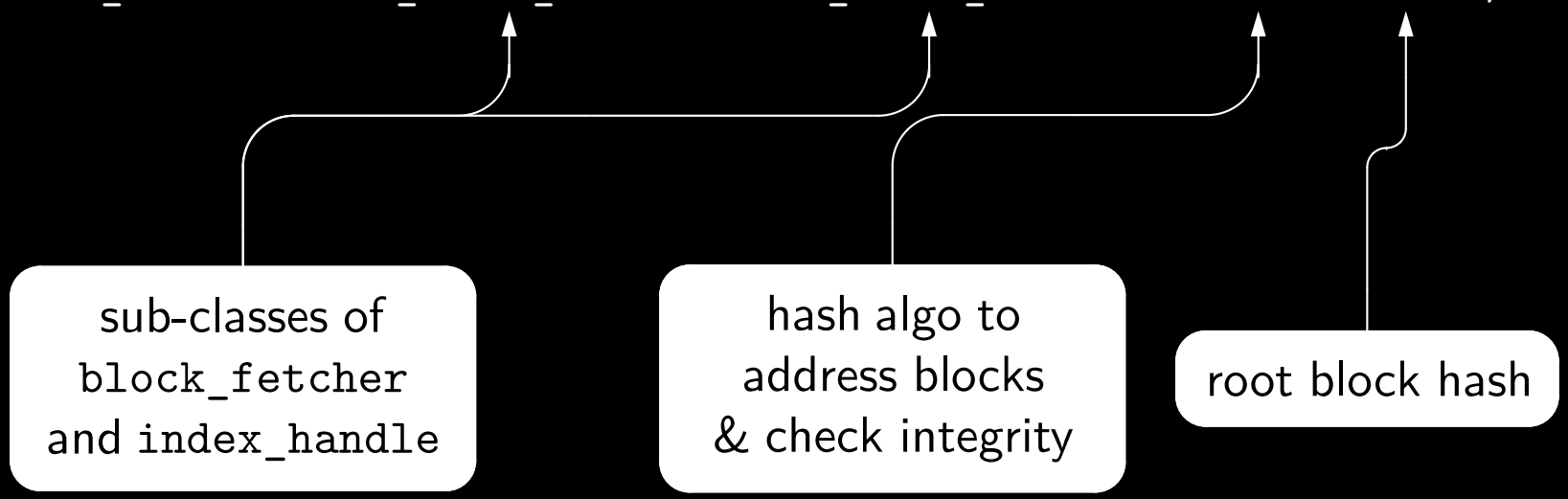


block store implementations

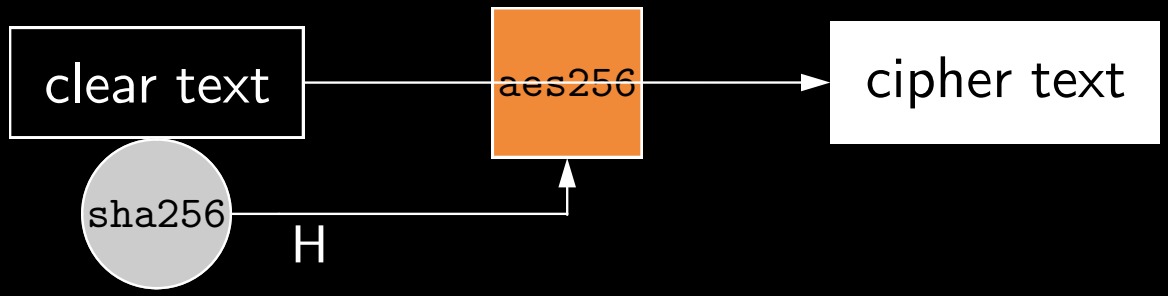
- gdbm, tdb, bdb, etc.
- file system (à la Git)
  - easily transferred over HTTP, FTP, rsync, etc.
- remote (`chop-block-server`)
  - ONC RPC, over TLS
  - discovery using Avahi (`chop-store-discover`)

block indexer: content-addressable storage

```
$ chop-archiver -i hash_block_indexer -I sha1 \
                --archive britney.mp3
tree_indexer:hash_block_fetcher:hash_index_handle:64:sha1:7c4…x7e/42
```



block indexer: content-hash keys




```
$ chop-archiver -i chk_block_indexer \
                -I aes256,cbc,sha256,sha1 \
                --archive naked.jpg
…:chk_index_handle:64:aes256,cbc,sha1:7c4…x7e,et4…vek/8a
```

cipher algo & integrity check

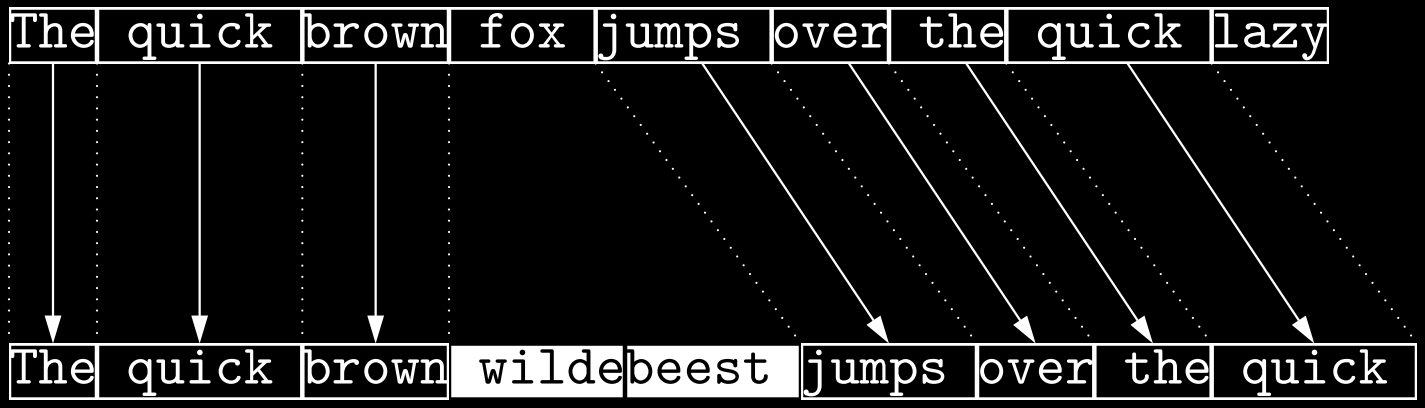
sha1(cipher-text)

key = sha256(clear-text)

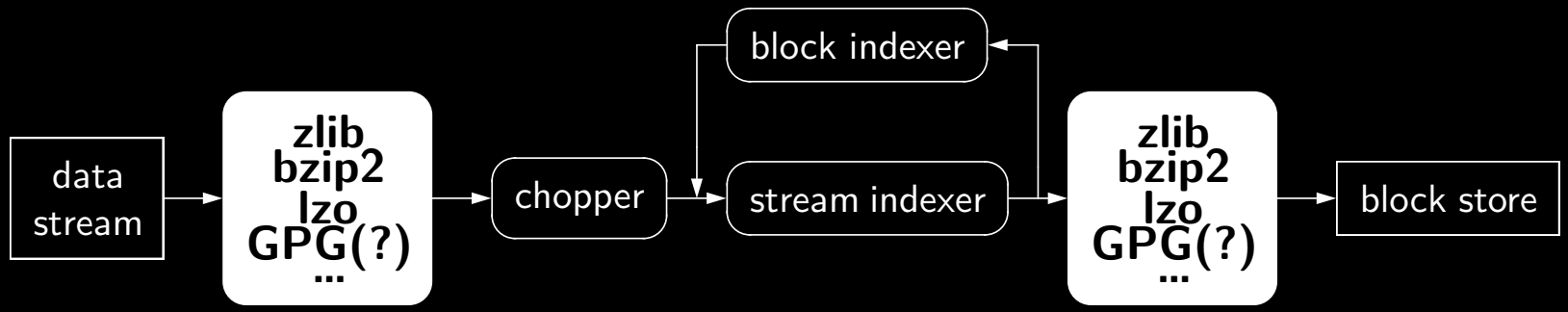
content-addressable storage & compression

fixed-size chopper

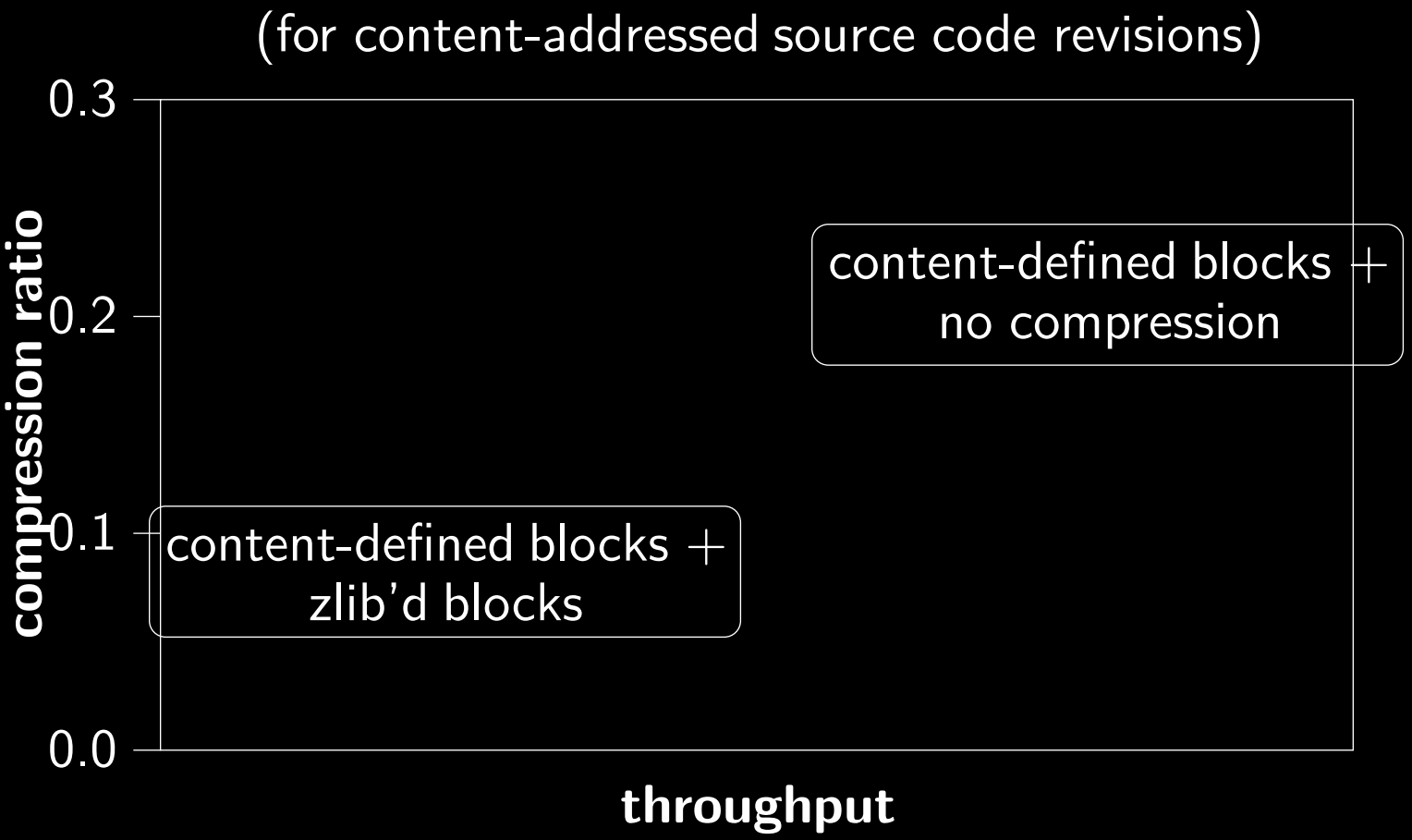
content-based chopper



filters

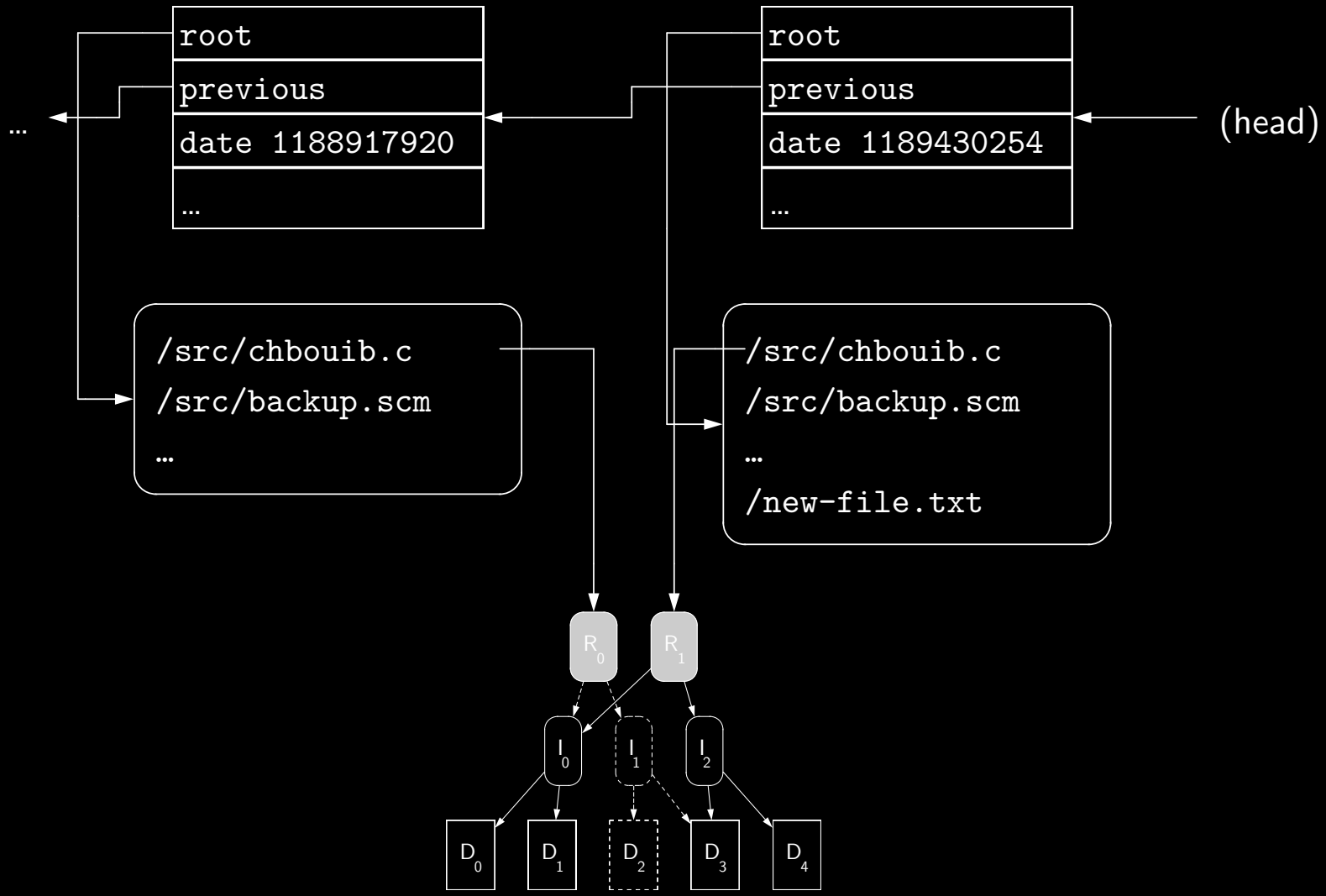

block indexer
data stream
zlib
bzip2
lzo
GPG(?)
...
chopper
stream indexer
zlib
bzip2
lzo
GPG(?)
...
block store

tradeoffs

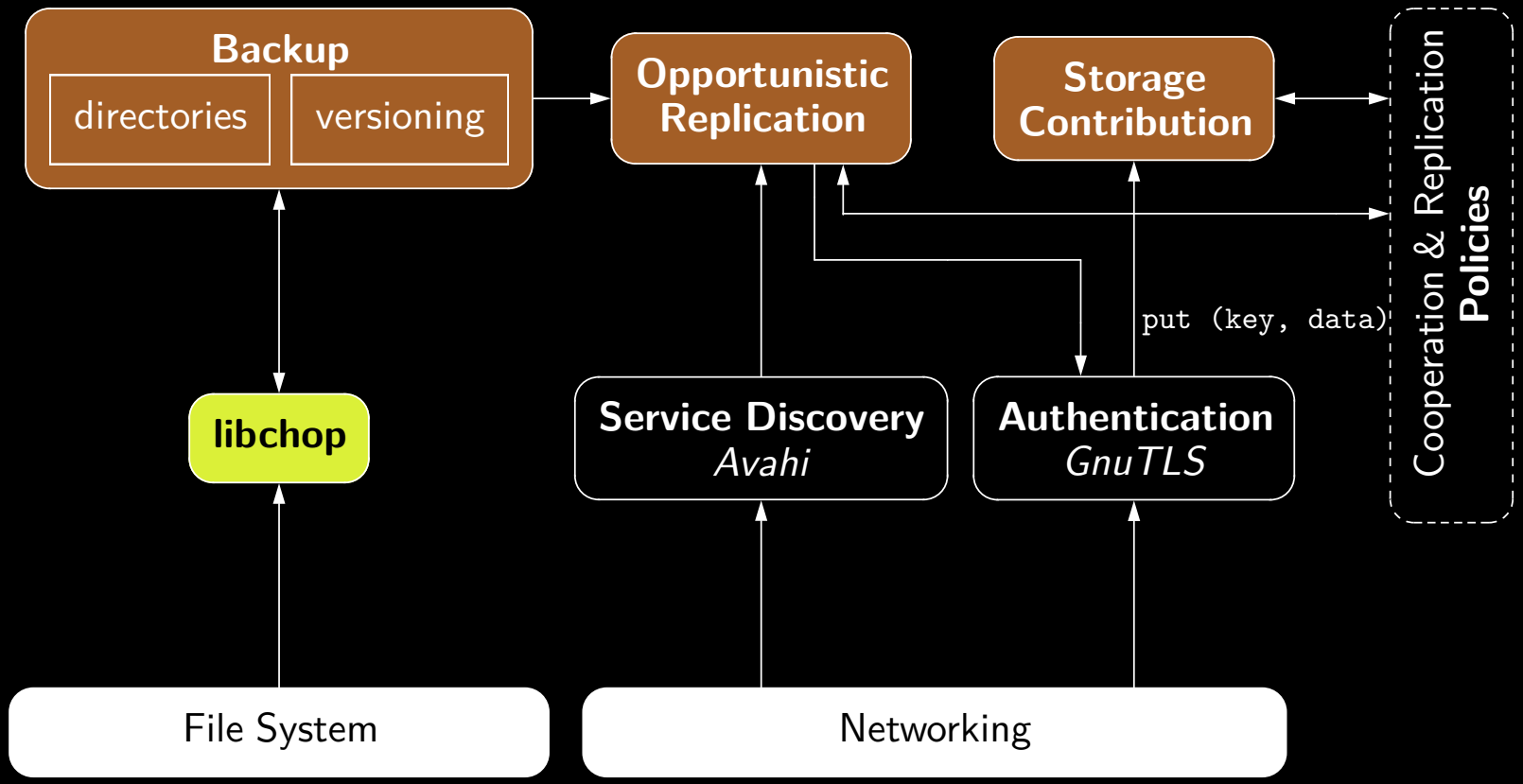

(for content-addressed source code revisions)
0.3
0.2
0.1
0.0
compression ratio
content-defined blocks +
no compression
content-defined blocks +
zlib'd blocks
throughput

- The Basics
- The Mechanics
- **The Plan**
  - (built-in) support for directories & versioning
  - a cooperative/p2p backup daemon
  - storage in GNU

(built-in) support for directories & versioning

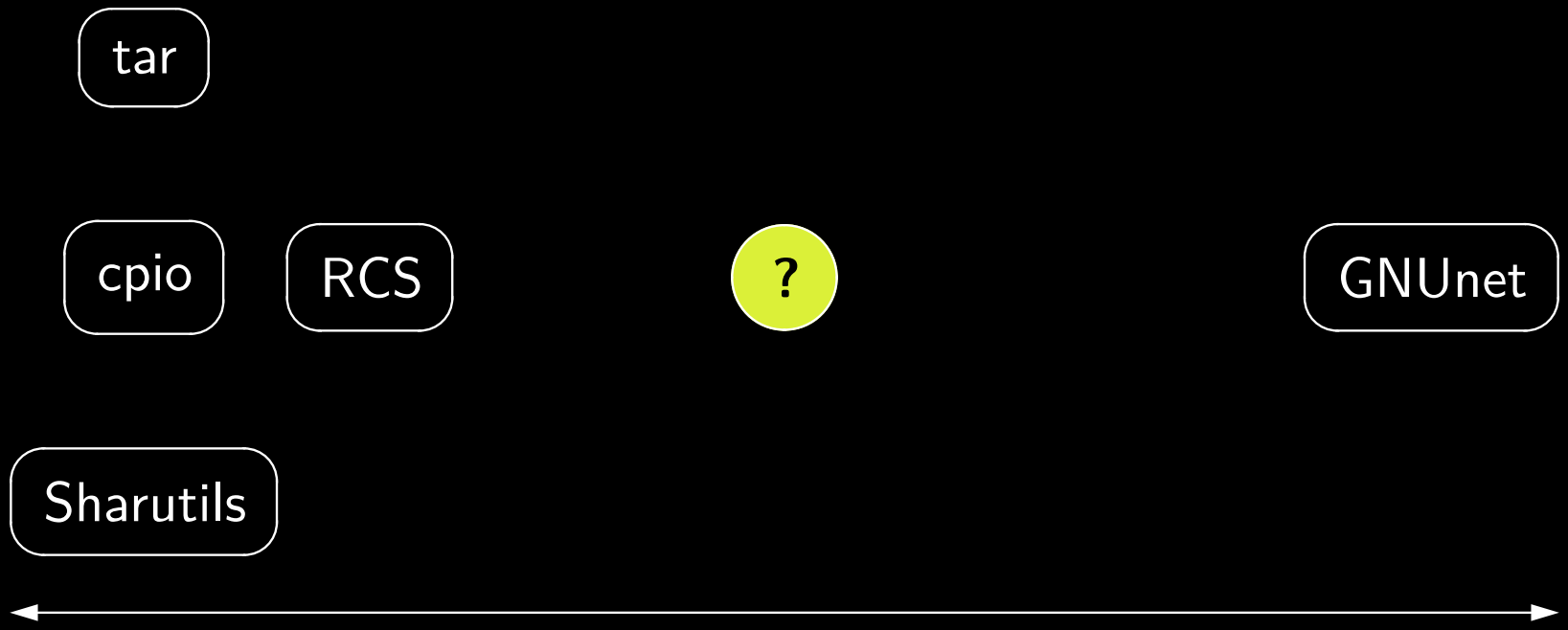


a cooperative/p2p backup daemon



your idea here

storage in GNU



Summary

libchop is cool!

Summary

- **mechanisms**, not policy
- content-addressable storage, content-hash keys, etc.
- supports **disseminated** data blocks
- build **distributed** backup/storage tools!

Thanks!

ludo@gnu.org

http://www.nongnu.org/libchop/