

Lossless-compressed data storage for SWH

Compressed, tunable & energy-aware

P. Ferragina, F. Tosoni



Department
of Excellence
2023 - 2027

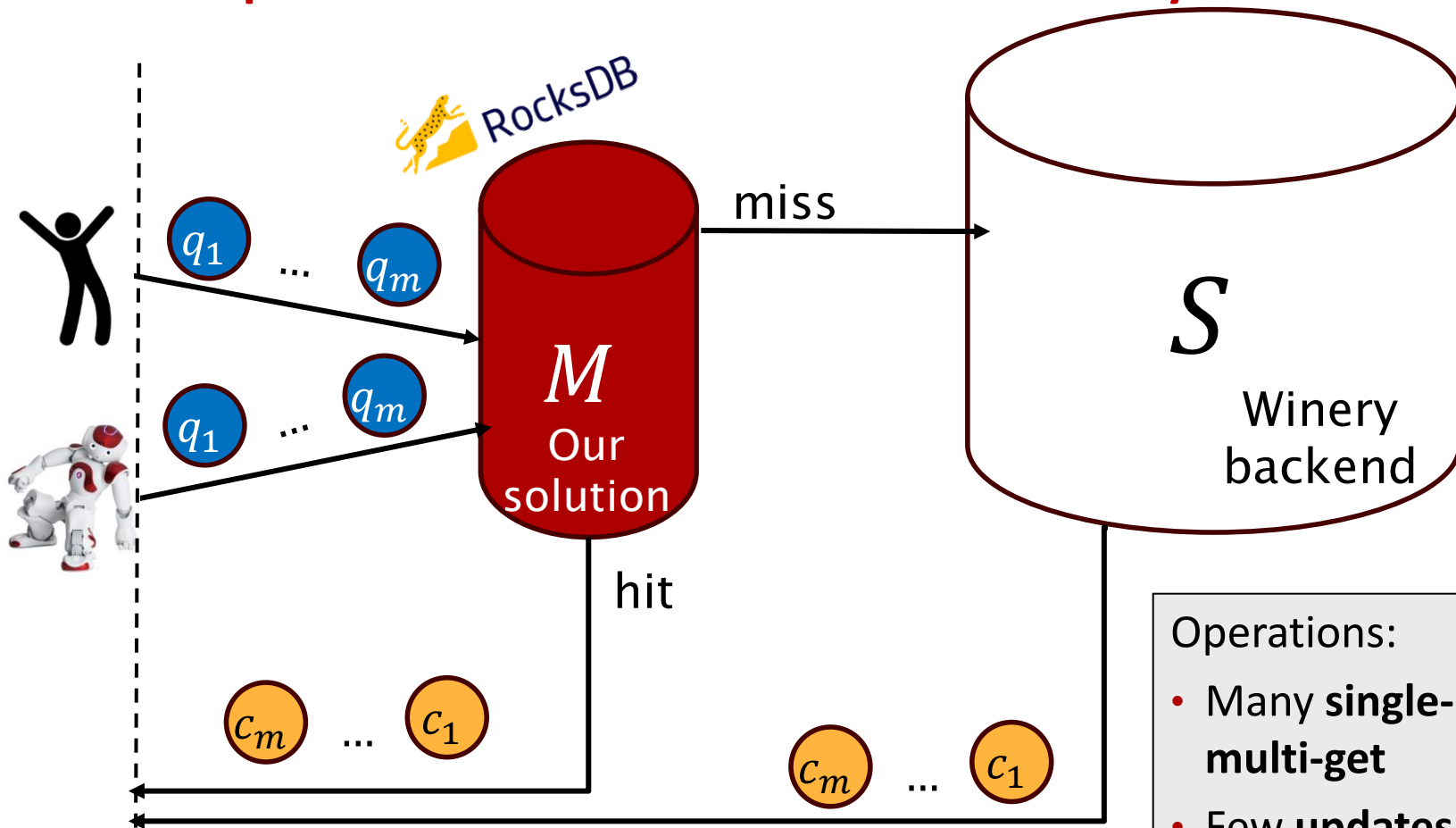
LEMbeDS
Economics, Management and
Law in the era of Data Science



Compressed cache based on a key-value store

Milestone 1

Compressed cache based on a key-value store



Given a stream q_1, \dots, q_m of queries
we aim to design a scalable and
compressed cache of size M

Operations:

- Many **single-** and **multi-get**
- Few **updates** (insertion, deletions)

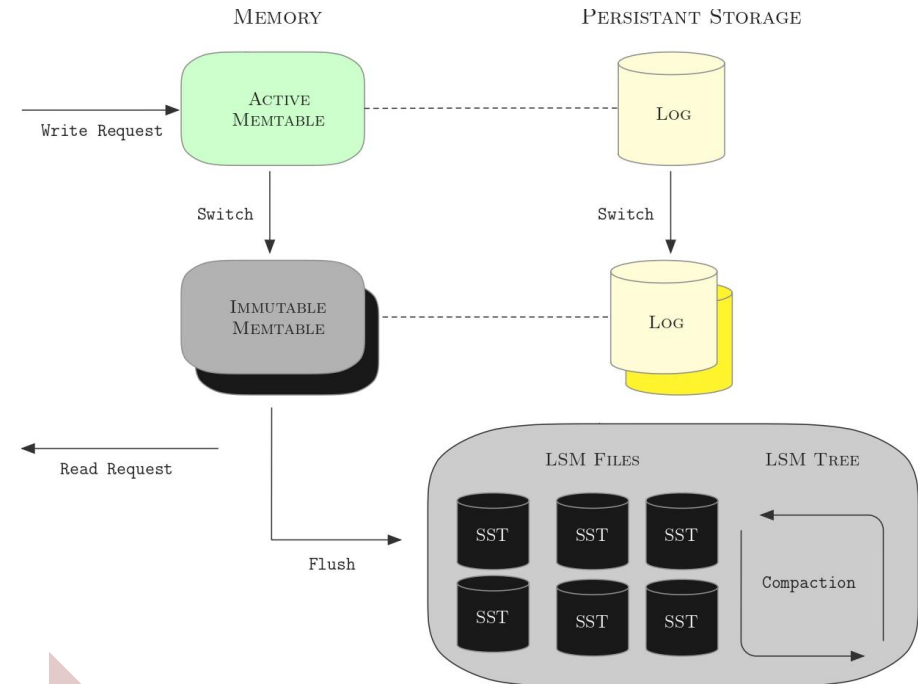
RocksDB



RocksDB  Meta

A fine-tunable **key-value store** based upon the **LSM-Tree**.

- Concurrency handling
- Single- & multi-get retrieval
- Fast insertion/delete



Ordered
by key

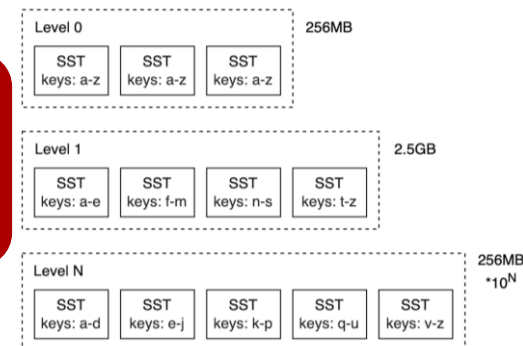
⇒ **P**ermute

Divided into
blocks

⇒ **P**artition

Compressed
blocks

⇒ **C**ompress



Repository features

pybind11



zlib

ppc-swh-rocksdb++ is a cmake/C++ NoSQL DB with Python Bindings.

Key features:

- Content-based key generation (Regex or TLSH hash).
- Benchmarking for compression ratio gains.
- Verification and inspection tools for parquets, tars, shards and more.

External dependencies:

- pybind11 for C++ to Python binding.
- Apache Arrow for the parquet format.
- jemalloc: memory allocator.
- Compression libraries: gflags, zstd, snappy, zlib.


RocksDB



GRENOBLE ALPES
RECHERCHE

INFRASTRUCTURE DE
CALCUL INTENSIF
ET DE DONNÉES

Experiments replicated on the
Perseus/GRICAD cluster
(Univ. of Grenoble).

 **Work in progress:** construct
an exhaustive Terabyte-scale
subset of The Stack v2 in
parquet format comprising
Python, Java, Javascript,
C/C++. (*discussed later*)

Some technicalities:

- Dependencies installed manually or via `nix`
- Tests conducted on write/read operations in multi-threaded settings
- Enabled Intel® RAPL energy profiler (by contacting the node admin)
- Tasks scheduled via OAR (this introduces some delay)

Which are the keys?

Context based

key =

Extension+filename+SWHID

```
py.BuildAggregatePS3Table | sw  
:1:cnt:238e0908f0180e279f02b8  
14186661508a0462d1
```

```
py.DeckMenu | sw:1:cnt:266433a  
578d4bd3bba74c9f9e8c33fe2869f  
a57b
```

```
java.AbstractBaseRedisDao | sw  
:1:cnt:03f261eca0fa67a7f6946f  
496eeb1dc352b17ea7
```

Which are the keys?

Context based

key =

Extension+filename+SWHID

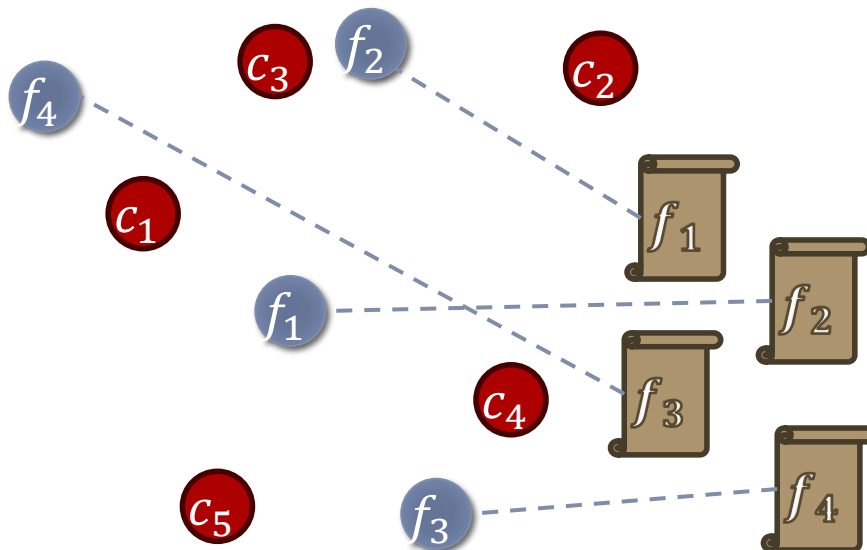
```
py.BuildAggregatePS3Table | sw  
:1:cnt:238e0908f0180e279f02b8  
14186661508a0462d1
```

```
py.DeckMenu | sw:1:cnt:266433a  
578d4bd3bba74c9f9e8c33fe2869f  
a57b
```

```
java.AbstractBaseRedisDao | sw  
:1:cnt:03f261eca0fa67a7f6946f  
496eeb1dc352b17ea7
```

Content based

- LSH-Project each document f_1, \dots, f_n in a vector space
- Relevant class and function names, comments, statements and more



Which are the keys?

Context based

key =

Extension+filename+SWHID

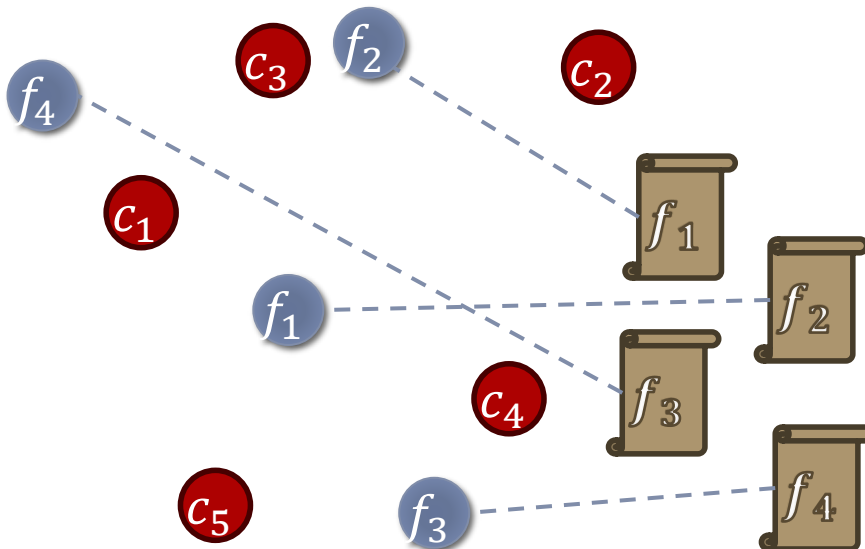
```
py.BuildAggregatePS3Table | sw  
:1:cnt:238e0908f0180e279f02b8  
14186661508a0462d1
```

```
py.DeckMenu | sw:1:cnt:266433a  
578d4bd3bba74c9f9e8c33fe2869f  
a57b
```

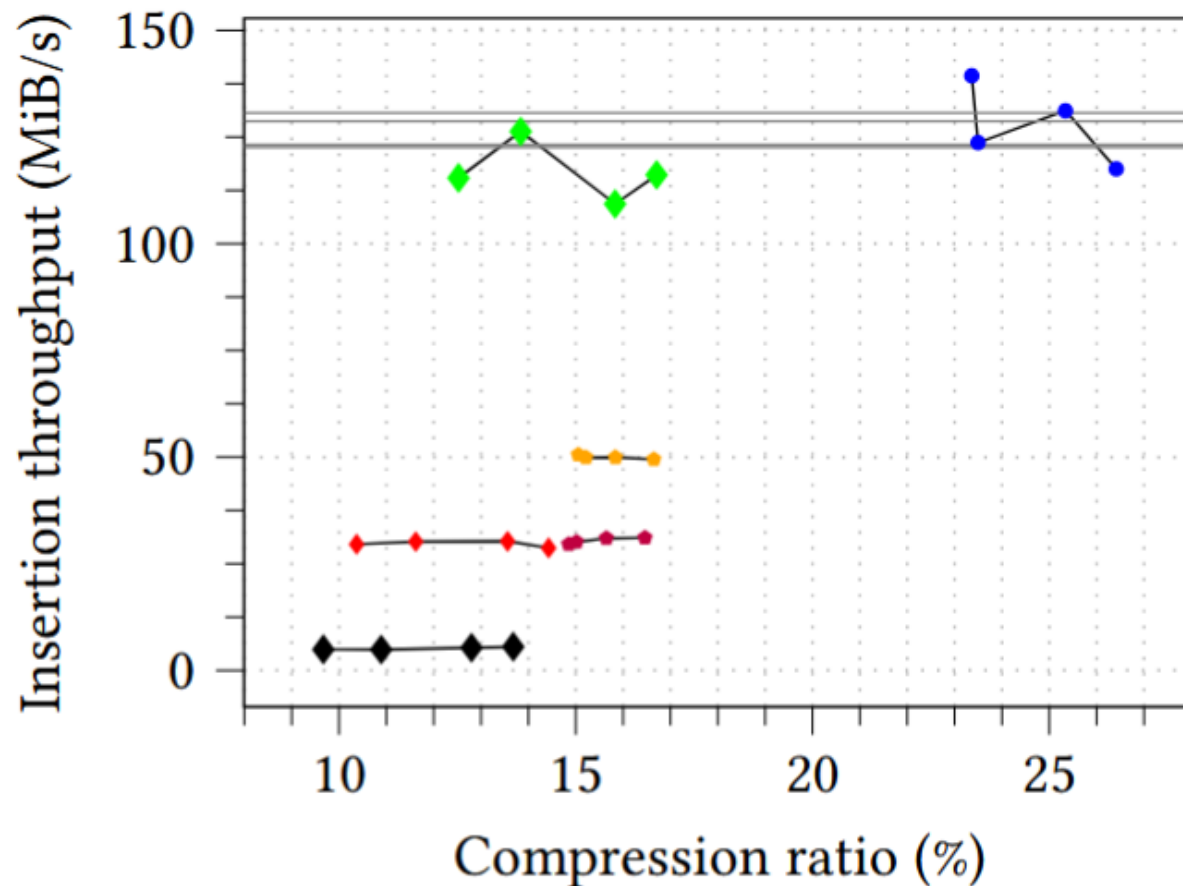
```
java.AbstractBaseRedisDao | sw  
:1:cnt:03f261eca0fa67a7f6946f  
496eeb1dc352b17ea7
```

Content based

- LSH-Project each document f_1, \dots, f_n in a vector space
- Relevant class and function names, comments, statements and more



Fine tuning: insertion

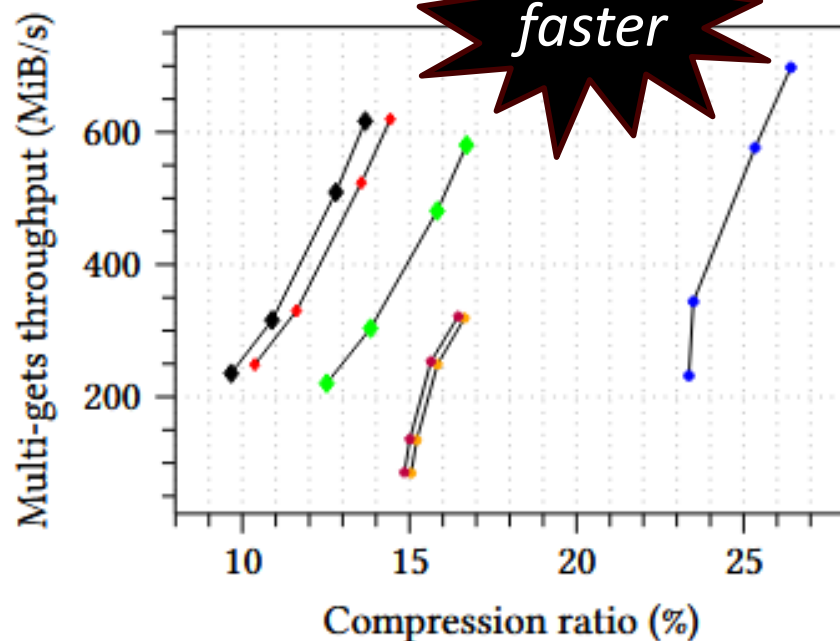
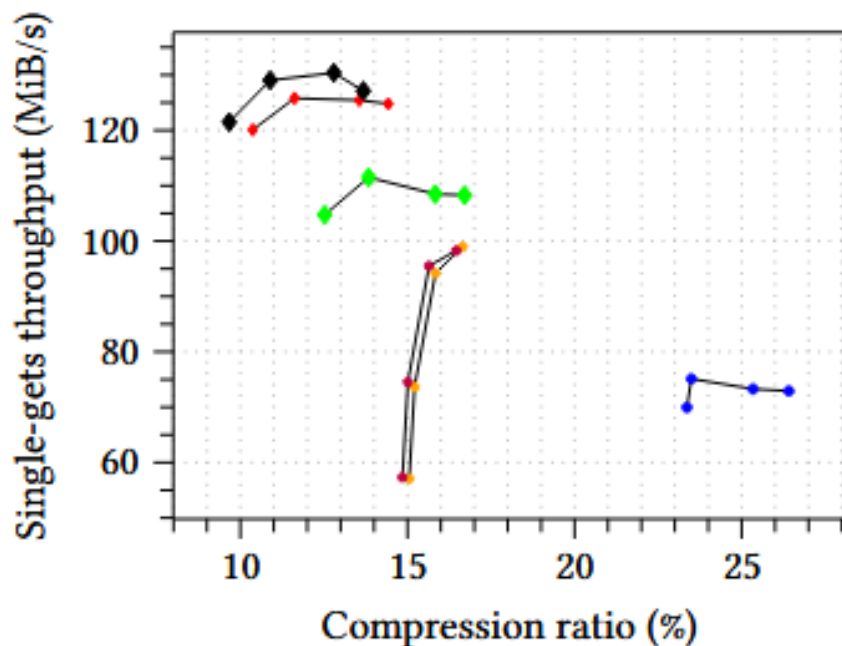


Fine tuning:
10 GiB
Python code



—◆— ZSTD-3 —◆— ZSTD-12 —◆— ZSTD-22 —◆— Zlib-6 —◆— Zlib-9 —◆— Snappy-0

Fine tuning: single- vs multi-get



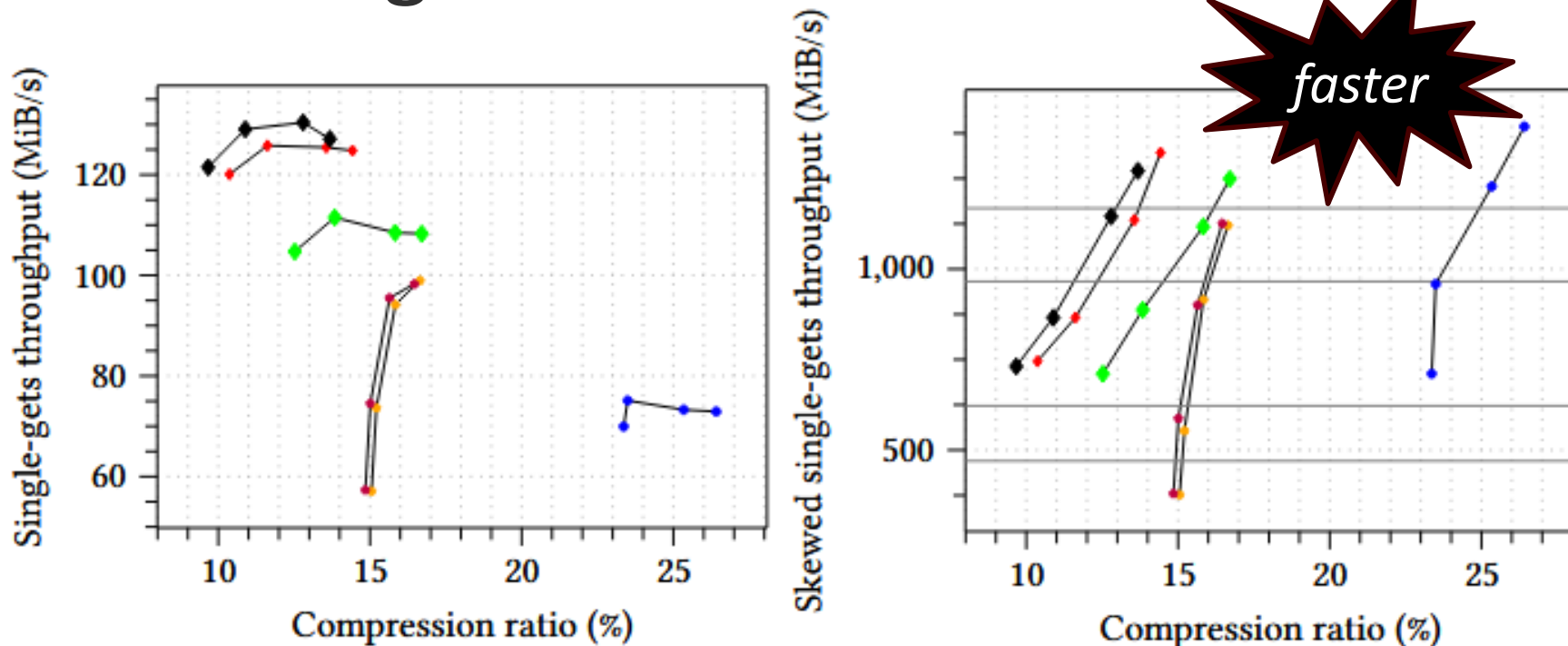
faster



- 6 compression algorithms/levels
- 4 block sizes: 4KiB, 16KiB, 64KiB, 128KiB
- 140 sequential code version tests

—◆— ZSTD-3 —◆— ZSTD-12 —◆— ZSTD-22 —◆— Zlib-6 —◆— Zlib-9 —◆— Snappy-0

Fine tuning: uniform vs skewed data

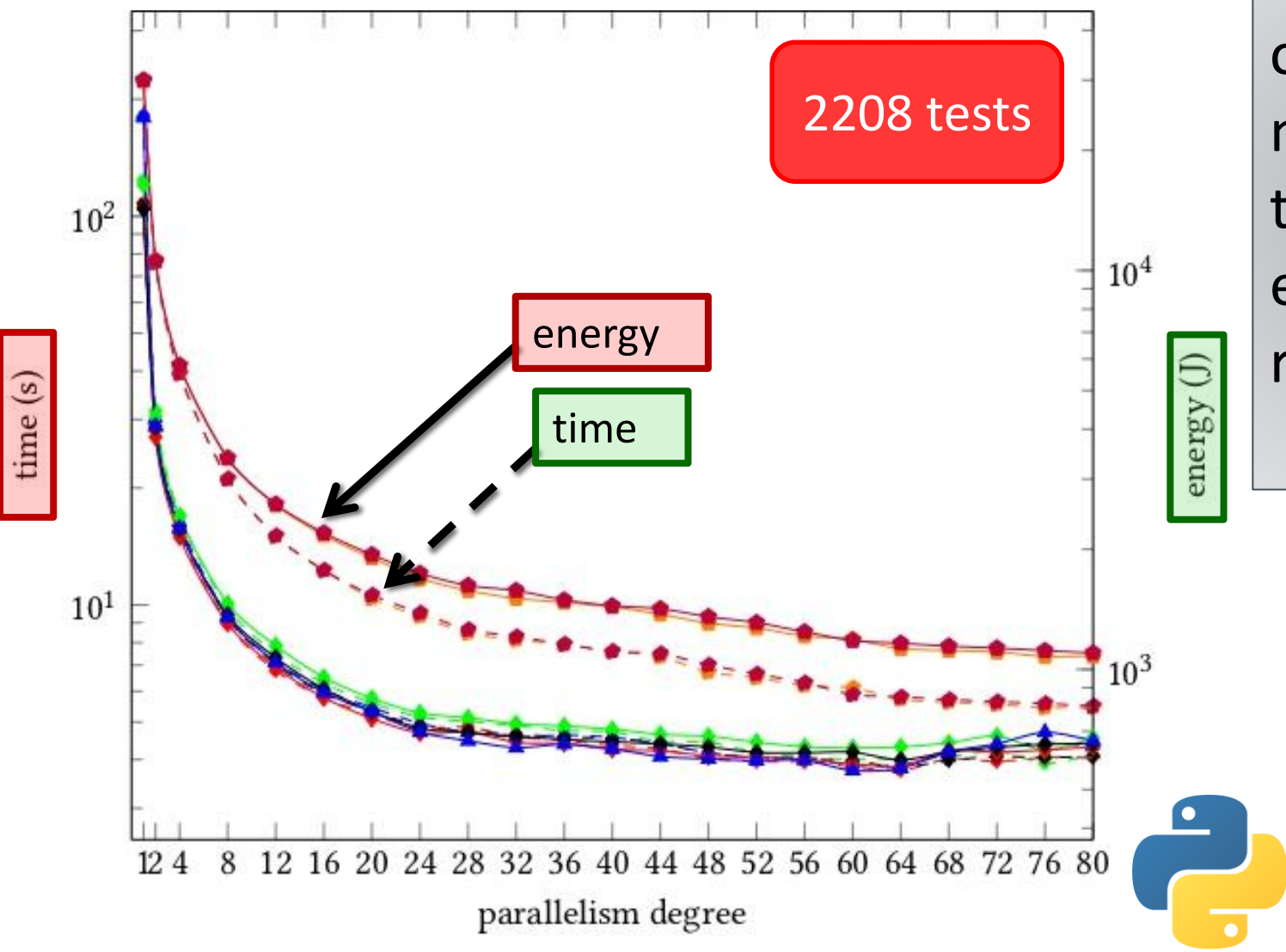


Power law, $\alpha = 1.5$

- 6 compression algorithms/levels
- 4 block sizes: 4KiB, 16KiB, 64KiB, 128KiB
- 140 sequential code version tests

—◆— ZSTD-3 —◆— ZSTD-12 —◆— ZSTD-22 —◆— Zlib-6 —◆— Zlib-9 —◆— Snappy-0

Stream-parallel & energy aware



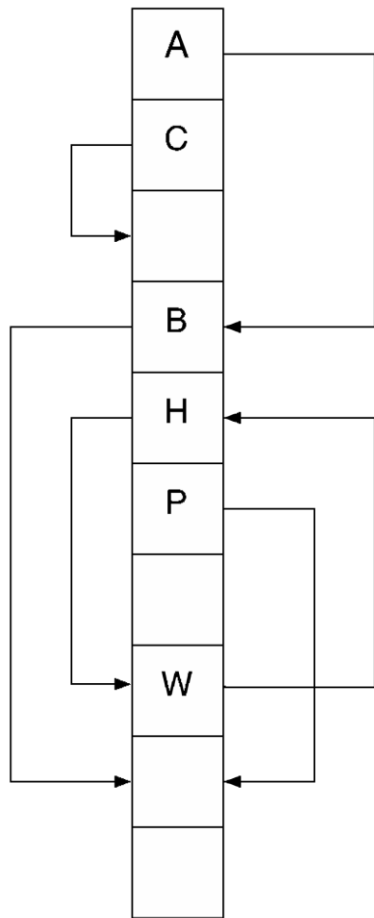
Green: 1-2 orders of magnitude time & energy reduction



- Zstd 3
- Zstd 12
- Zstd 22
- Zlib 6
- Zlib 9
- Snappy

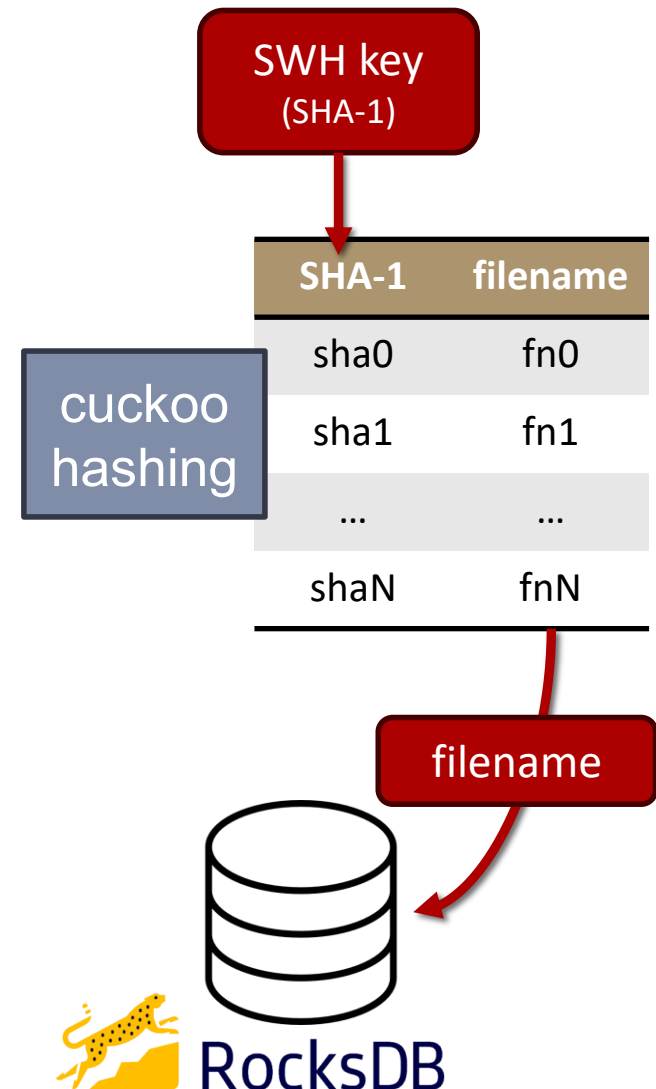


What if we need to keep the SWHID?



Cuckoo hashing
(`libcuckoo`) for a
space-efficient and
dynamic mapping
SHA-1→filenames.

We might consider
replacing cuckoo
hashing with an
additional RocksDB
instance as the
number of entries
grows.



Conclusions



Our **PPC** approach hinging on **RocksDB** offers...

- Dynamic, re-configurable and performant solution
- Competitive compression (~12% on 10 GiB, we expect more scaling up)
- Fast **insertion**/deletion (>100 MiB/s for zstd-3)
- Fast random **access**
 - **Single-gets**: >100 MiB/s (zstd-3), up to 140 MiB/s (zstd-22)
 - **Multi-gets**: ~600 MiB/s (zstd-3)
- Saves 1-2 orders of magnitude time-energy via parallelism
- Remarkable retrieval speedup as thread count increases (~9 GiB/s, zstd-3)

Efficient caching system for **Winery**, **Code2Code** search engines

Milestone 1: Goals & Roadmap

Formatting Python, Javascript, Java, C/C++ code sources in parquet format.

the-stack-v2-pathsliced



- content
- .tar.zst

the-stack-v2-train-full-files/data



- content + metadata
- parquet

To avoid write amplification problems, we transferred the datasets on UniPi machines for further elaboration



the-stack-v2/data/Python

- metadata
- parquet

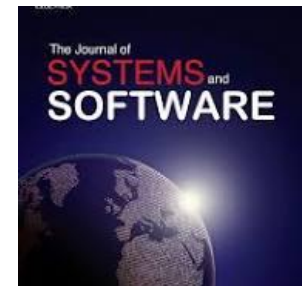
C++
Javascript
Java

Milestone 1: Goals & Roadmap

Finalise the experiments on the Terabyte-scaled dataset \Rightarrow contributing a dynamic and tunable storage engine.

Our narrative: a lossless-compressed and dynamic, Terabyte-scale, energy-aware, highly-tuned caching solution based on PPC paradigm and a NoSQL DB.

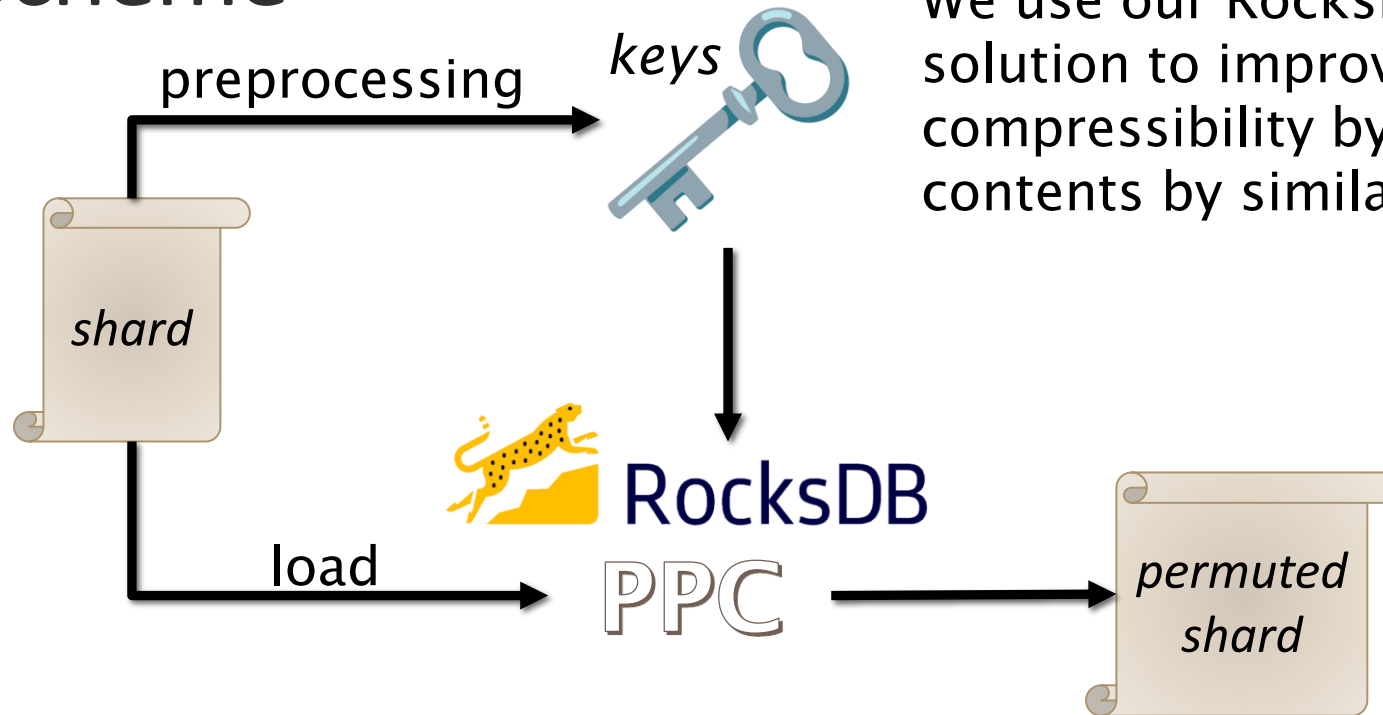
Compared to Boffa *et al.* [2025], our solution is dynamic and highly tunable, achieves more robust and performant results especially for the retrieval phase.



Permuting SWH shard files by content similarity

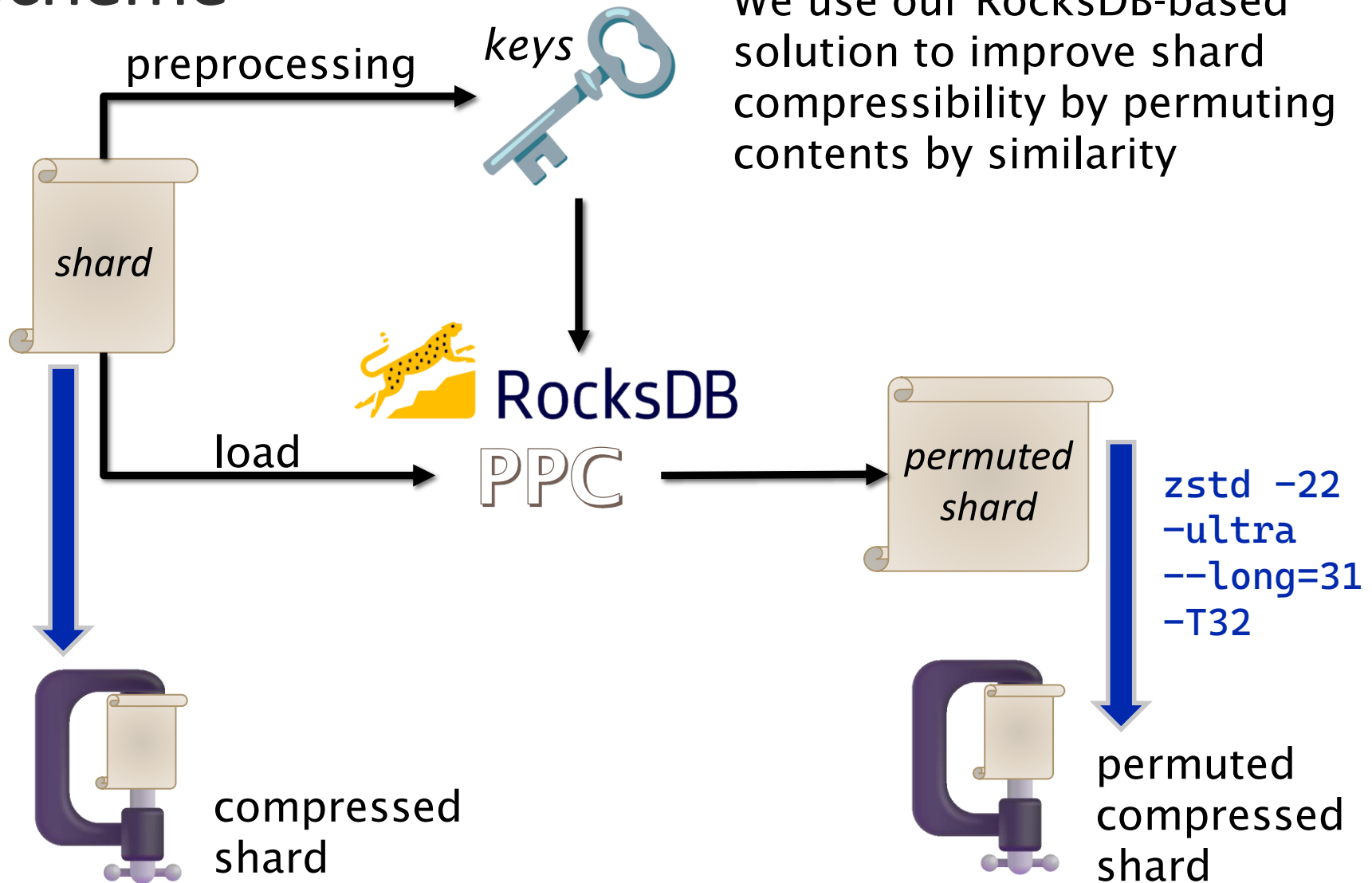
Milestone 2

Scheme

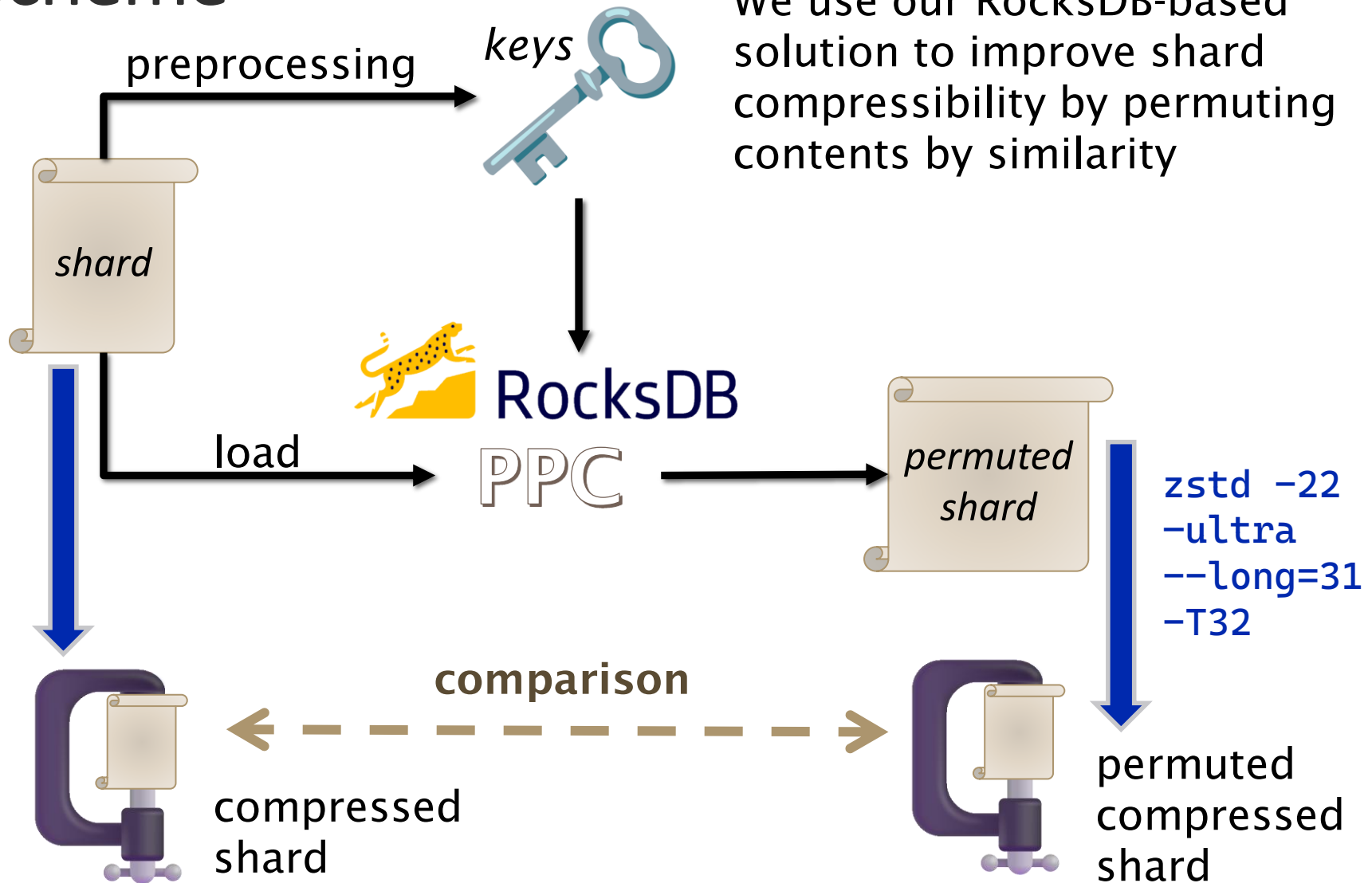


We use our RocksDB-based solution to improve shard compressibility by permuting contents by similarity

Scheme



Scheme



Regex

The critical step is defining and computing a proper sorting. Context-based information is not available, so we shall consider *content-based* ones.

Options:

- LSH computations (e.g., via minhash or TLSH). \Rightarrow **time-expensive: 19h with ~50 cores and 2 permutations.**
- An alternative approach entails extracting multi-language class & function names, comments and more. \Rightarrow **faster**

```
# Define the comprehensive regular expression (as provided in your original code)
COMPREHENSIVE_NAME_REGEX = re.compile(
    '<title>(.*?)</title>|'           # HTML title tag (ORA PRIMO)
    'class\s+(\S+):|'               # Python class
    'def\s+(\S+)\s*(.*?)|'          # Python function
    'class\s+(\S+)\s*(?:extends\s+\S+)?(?:\s+implements\s+\S+(?:,\s*\S+)*)?\s*{|' # Java/C++ class
    'function\s+(\S+)\s*(.*?)\s*{|' # JavaScript named function
    '<h[1-6]>(.*?)</h[1-6]>|'       # HTML heading tags (h1-h6)
    '<[^>]*\sid=[\"\\']?([\\w:-]+)[\"\\']?[>]*>|' # HTML id attribute
    '<a\s+href=[\"\\']?(\\S+)[\"\\']?[>]*>|' # HTML href attribute
```

A standalone for permuting

The permuted shard inherits most portions of the input shard.

	offset	length
SHARD_MAGIC	0	SHARD_OFFSET_MAGIC (32)
header	32	(56)
version		uint64_t (8)
objects_count		uint64_t (8)
objects_position (op)		uint64_t (8)
objects_size		uint64_t (8)
index_position (ip)		uint64_t (8)
index_size		uint64_t (8)
hash_position (hp)		uint64_t (8)
Objects	<op>	
object0 size		uint64_t (8)
object0 data		<object0 size>
object1 size		uint64_t (8)
object1 data		<object1 size>
...		
Index	<ip>	
object0 key		SHARD_KEY_LEN (32)
object0 offset		uint64_t (8)
...		
Hash map	<hp>	
hash function		<as written by cmph_dump>

Permute
the objects

Update the
offsets

NEW

Milestone 2: Goals & Roadmap

- Goal: a significant infrastructure contribution for a dedicated column/section in an **infrastructure white paper**.
- We aim to demonstrate the feasibility of **reducing shard space** by ~5% in the compressed domain \Rightarrow petabyte scale.
- *Optional*: Showcase reduced **energy** consumption for block-accesses on compressed shard files.



Q&A

Francesco Tosoni, PhD
Postdoctoral researcher

Informatica
L.go B. Pontecorvo 3
56127 Pisa PI
Italia

francesco.tosoni@di.unipi.it
pages.di.unipi.it/tosoni

