Compressed data structures for big data indexing

Giulio Ermanno Pibiri

Department of Environmental Sciences, Informatics and Statistics

@giulio_pibiri

@jermp

Campus TALES, 12 July 2024

Myself

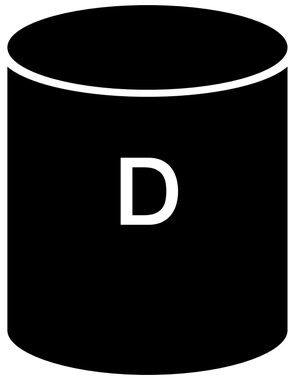
- **Giulio Ermanno Pibiri**
- PhD in Computer Science from the University of Pisa in 2019
- As of June 2022, RTD-B at Ca’ Foscari, DAIS
- Personal Web page: https://jermp.github.io
- GitHub: https://github.com/jermp

Compressed data structures

- **Research**: design compressed data structures to index large quantities of data.

Compressed data structures

- **Research**: design compressed data structures to index large quantities of data.

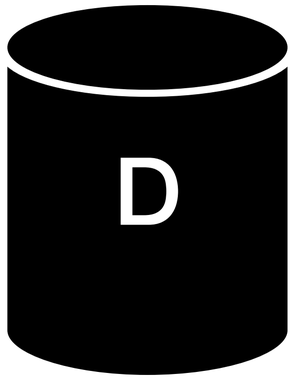


500 GB

- Given a query Q, how to answer Q on D as efficiently as possible?
- You **cannot** just scan D: too slow.
  (And if you have millions of such queries per day to answer?)

Compressed data structures

- **Research**: design compressed data structures to index large quantities of data.



500 GB

- Given a query Q, how to answer Q on D as efficiently as possible?
- You **cannot** just scan D: too slow.
  (And if you have millions of such queries per day to answer?)
- **Solution**: pre-process D into a **data structure**.

  Two-fold objective:
  1. **reduce the storage space** for D;
  2. make queries **fast**.

Compressed data structures

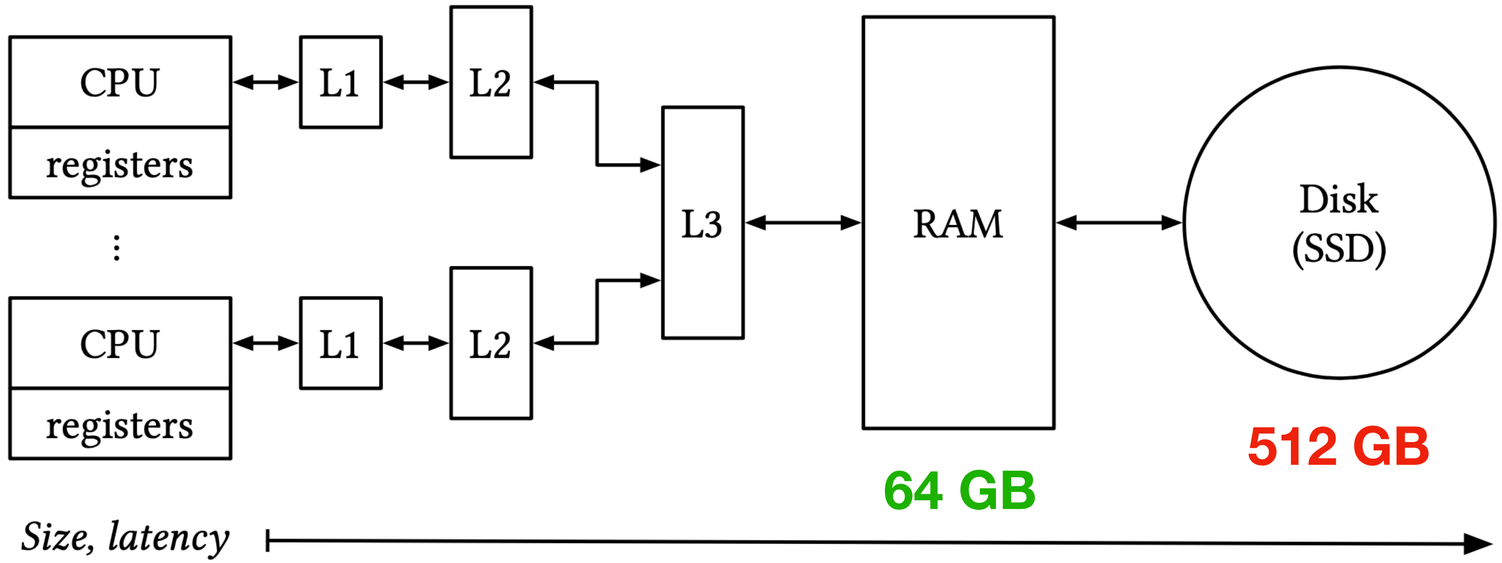
A compressed data structure uses **less storage space** than the original data, thus permitting:

- for a fixed memory budget, to **handle larger datasets**;

Compressed data structures

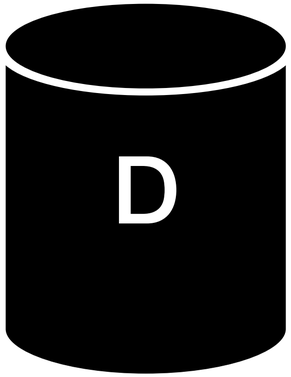
A compressed data structure uses **less storage space** than the original data, thus permitting:

- for a fixed memory budget, to **handle larger datasets**;
- for the same dataset, to maintain its compressed representation in **faster memory levels**.



RAM is orders of magnitude faster than the disk!

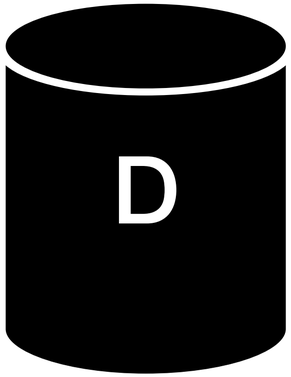
An example



150,000 *S. Enterica* genomes

5 MB for genome → **750** GB
1.5 MB per genome if compressed with `gzip` → **225** GB

An example



Q = “ *In what genomes, does*

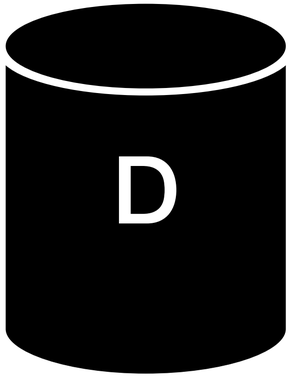
`GTTGGGCGGCCCCTTCGGTTGGGCCAAAGATCTTCAGACCCGCCGC`

appear ? ”

150,000 *S. Enterica* genomes

5 MB for genome → **750** GB
1.5 MB per genome if compressed with `gzip` → **225** GB

An example



Q = “ *In what genomes, does*

`GTTGGGCGGCCCCTTCGGTTGGGCCAAAGATCTTCAGACCCGCCGC`

appear ? ”

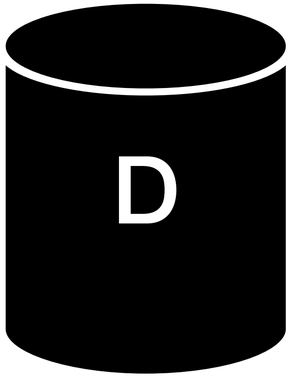
150,000 *S. Enterica* genomes

5 MB for genome → **750** GB
1.5 MB per genome if compressed with `gzip` → **225** GB

the Fulgor index (2023): **71** GB

https://github.com/jermp/fulgor

An example



Q = “ *In what genomes, does*

`GTTGGGCGGCCCCTTCGGTTGGGCCAAAGATCTTCAGACCCGCCGC`

appear ? ”

150,000 *S. Enterica* genomes

5 MB for genome → **750** GB
1.5 MB per genome if compressed with `gzip` → **225** GB

the Fulgor index (2023): **71** GB

https://github.com/jermp/fulgor

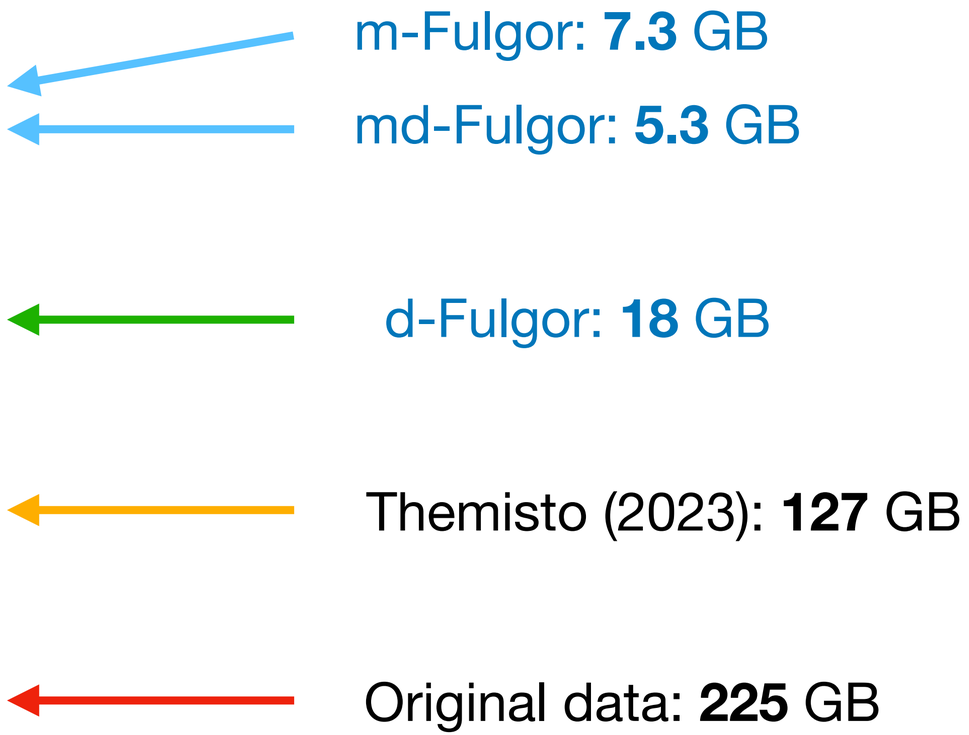
new results (2024):

d-Fulgor: **18** GB
m-Fulgor: **7.3** GB
md-Fulgor: **5.3** GB

Save money!

Amazon EC2 instances pricing:

- https://instances.vantage.sh/aws/ec2/x2gd.medium
  16 GiB of RAM — **73 $** per month
- https://instances.vantage.sh/aws/ec2/x2gd.xlarge
  64 GiB of RAM — **292 $** per month
- https://instances.vantage.sh/aws/ec2/x2gd.2xlarge
  128 GiB of RAM — **584 $** per month
- https://instances.vantage.sh/aws/ec2/x2gd.4xlarge
  256 GiB of RAM — **1168 $** per month



Numbers taken on 10/07/2024.

Inverted indexes

You have a large collection of Web pages, like several millions.

Problem: Given *k* words, how to find all Web pages where these words occur?

RDF Triples indexing

You have a large collection of RDF triples (S,P,O), like 350 millions.

Problem: Given a **wildcard query** like (? ? O) or (? P ?), how to return **all** matching triples?

Language models

amazon alexa

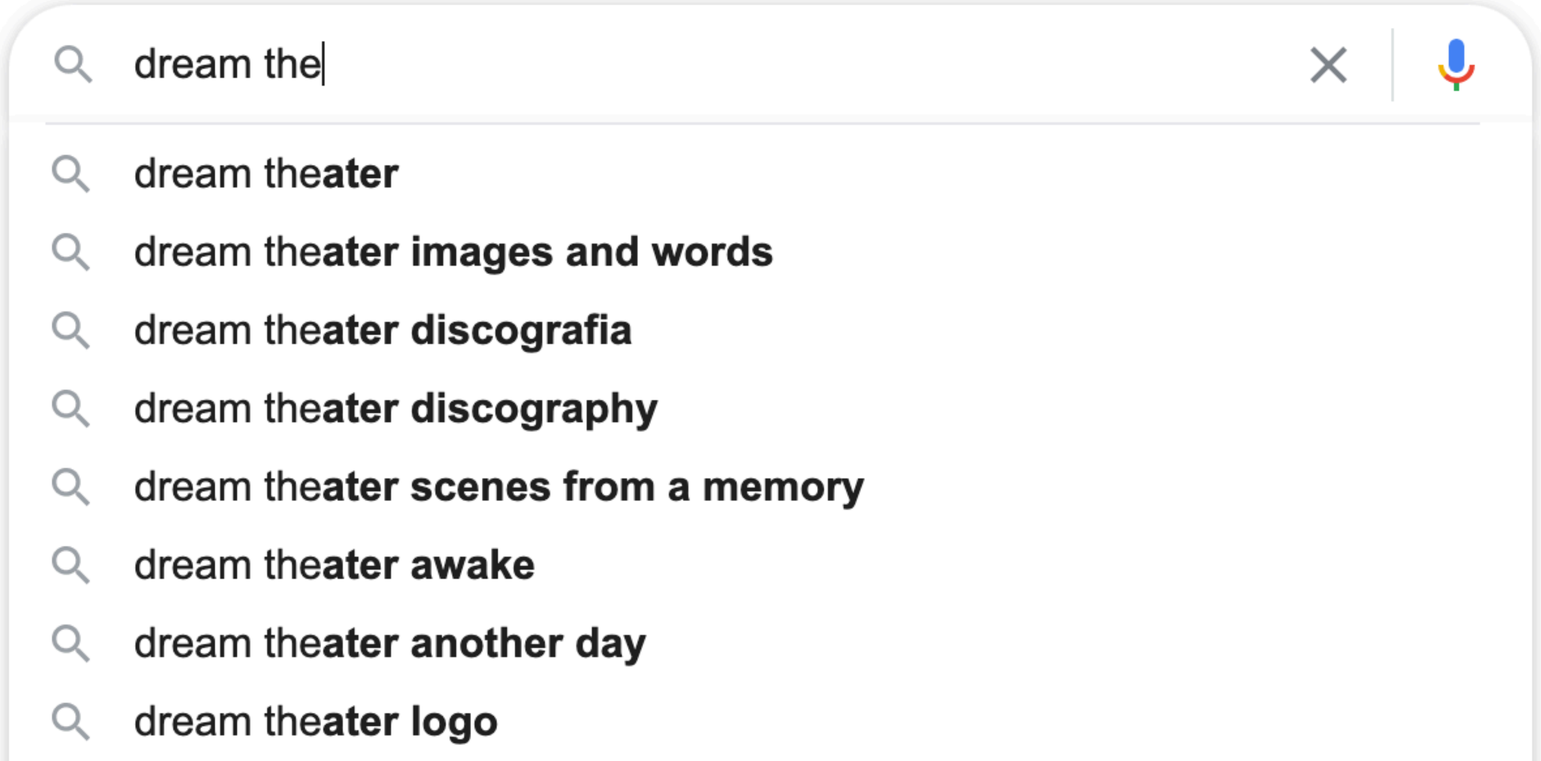
Google Translate

You have a large collection of q-grams, like 11 billions (the “Google-books” collection).

Problem: How, given a q-gram, return its context probability as fast as possible?

Query auto-completion

Problem: Given a collection S of scored strings and a partially completed user query Q, how to find the top-*k* strings that "match" Q in S?



Conclusions

- Efficiency is the key to:
  - **build better applications/services** in terms of reduced latency to access information (enhanced user experience);
  - **save computer resources** (power and storage machines).

Conclusions

- Efficiency is the key to:
  - **build better applications/services** in terms of reduced latency to access information (enhanced user experience);
  - **save computer resources** (power and storage machines).
- If you have to make some large-scale analysis and your computer gets stuck (i.e., slow execution or runs out of memory), get in touch!

Conclusions

- Efficiency is the key to:
  - **build better applications/services** in terms of reduced latency to access information (enhanced user experience);
  - **save computer resources** (power and storage machines).
- If you have to make some large-scale analysis and your computer gets stuck (i.e., slow execution or runs out of memory), get in touch!
- Other people involved: prof. **Nicola Prezza**, https://nicolaprezza.github.io.

Conclusions

- Efficiency is the key to:
  - **build better applications/services** in terms of reduced latency to access information (enhanced user experience);
  - **save computer resources** (power and storage machines).
- If you have to make some large-scale analysis and your computer gets stuck (i.e., slow execution or runs out of memory), get in touch!
- Other people involved: prof. **Nicola Prezza**, https://nicolaprezza.github.io.

Thank you for the attention!