Large-scale compression of software source code

A. Boffa, P. Ferragina
(joint also with A. Guerra, G. Manzini, G. Vinciguerra)

A³ lab: Advanced Algorithms and Applications
Dipartimento di Informatica
Why data compression?

It provides rich dividends:

• **Space** saving, of course!
  • ... and thus, reduced *storage and transmission* costs

• **Energy** saving: less servers involved

• **Speed**: not so obvious!
  • better exploitation of the *memory hierarchy*
  • reduced *burning out* of sectors in SSD disks
  • no need to *decompress* the whole data, in some approaches
Brotili: A General-Purpose Data Compressor

Authors:
- Jyrki Alakuijala
- Andrea Favruggia
- Paolo Ferragina
- Eugene Kliuchnikov
- Robert Obryk
- Zoltan Szabadka
- Lode Vandevenne

Published in:
- Journal
  ACM Transactions on Information Systems (TOIS) TOIS Homepage archive
Volume 37 Issue 1, December 2018 Issue-in-Progress
Article No. 4
ACM New York, NY, USA
table of contents doi> 10.1145/3231935
Searching Compressed Data

We were the first in 2000 to show how to search compressed data, without decompressing all of them.

Fast and accurate short read alignment with Burrows–Wheeler transform
H Li, R Durbin - bioinformatics, 2009 - academic.oup.com
Motivation: The enormous amount of short reads generated by the new DNA sequencing technologies call for the development of fast and accurate read alignment programs. A first generation of hash table-based methods has been developed, including MAQ, which is …

Cited by 17019

Ultrafast and memory-efficient alignment of short DNA sequences to the human genome
B Langmead, C Trapnell, M Pop… - Genome …, 2009 - genomebiology.biomedcentral.com
Bowtie is an ultrafast, memory-efficient alignment program for aligning short DNA sequence reads to large genomes. For the human genome, Burrows-Wheeler indexing allows Bowtie to align more than 25 million reads per CPU hour with a memory footprint of approximately…

Cited by 13275
The Software Heritage...

Size

As of today the archive already contains and keeps safe for you the following amount of objects:

Source files

13,841,373,660

Commits

2,886,851,936

Raw space (files) is about 1 PetaByte \( (10^{15} \text{ bytes}) \)
How to approach the problem?

Squash Compression Benchmark

The Squash library is an abstraction layer for compression algorithms, making it trivial to switch between them... or write a benchmark which tries them all, which is what you see here!

The Squash Compression Benchmark currently consists of 28 datasets, each of which is tested against 29 plugins containing 46 codecs at every compression level they offer—the number varies by codec, but there are 235 in total, yielding 6,580 different settings. The benchmark is currently run on 9 different machines for a current grand total of 59,220 configurations, and growing.
Three main families of compressors

[1977] Lempel-Ziv parsing and its derivatives

- The classic approach, made famous by gzip
- A revamped interest around it: 7z, zstd, brotli, lzfse, ...


- Better entropy-bounds than gzip-like compressors (bzip)
- But building a big BWT is costly and complicated...

[2000] Compressed (self-)indexes: offer space/decompress trade-offs

- **FM-index**, CSA, LZ-trie,... [Ferragina-Manzini, J.ACM ‘05]
  - In the compressed space, they include text + (full-text) index
  - Query/Decompression time is theoretically optimal

New compressors are based on sophisticated ML models, but slow.
Something to be careful…

We are talking about compressing a:
✔ collection of files of different types.

We have to apply a different compression scheme:

**PPC: Permute + Partition + Compress**

and study it in two experimental scenarios

- **BackUp** for the streaming access to compressed data
- **Random access** to single compressed files

Two main issues to deal with:
- Permuting based on which information? filename, path, file type, content,…
- Partitioning based on which block size? BackUp/Random access, compressor,..
Our experimental framework

The C and Python repositories in Github with most stars, each consisting of about 25 Gb.
BackUp scenario: Python files ≈ 25Gb

For random access, even with 4 MB blocks, HYBRID is very robust by losing only about 1% in compression ratio wrt Backup scenario.

On C files we get about 6% vs 27% compr. ratio of gzip
Random access: C files ≈ 25Gb

- Use blocks of 4-16 MBs

- The compression loss is about 1%
- The compression loss is about 3%
- The compression loss is about 10%

The difference on Python between `hybrid` and `filename_sorted` is negligible
The moral...

We compared Data-aware-PPC versus Context-aware approaches (à la GitPack), achieving the following results:

- Our features subsume the "context" at a larger decompression speed than GitPack
- They are robust with respect to block size
- zstd is a good compressor: large window at 100MB/sec speed

Next, we aim at:

- Investigating other orderings and partitionings
- C++ coding & HPC is needed because a single machine would take about 3 months of computation (≈ 200 Tb space at about 4k euro).

It’s time to try it on the whole SH