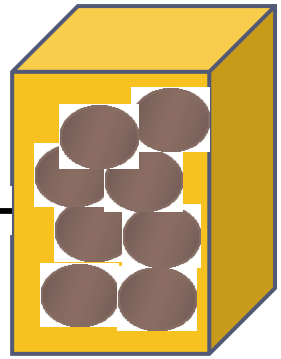
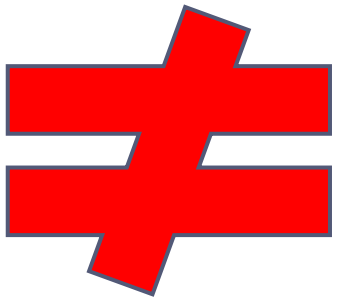
SDGen: Mimicking Datasets for Content Generation in Storage Benchmarks

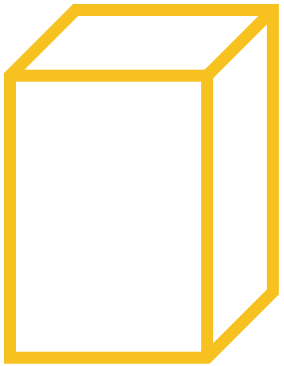
Raúl Gracia-Tinedo (Universitat Rovira i Virgili, *Spain*)
Danny Harnik, Dalit Naor, Dmitry Sotnikov (IBM Research-Haifa, *Israel*)
Sivan Toledo, Aviad Zuck (Tel-Aviv University, *Israel*)

Pre-Introduction

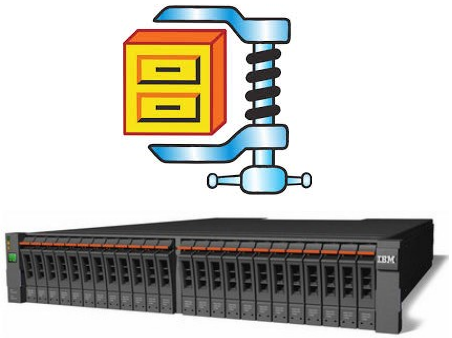


Stones in the backpack!!!

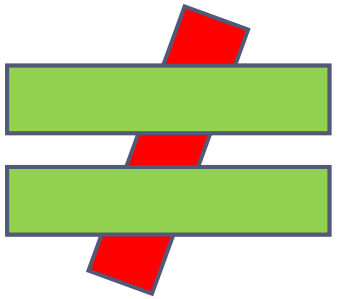




Just thin air ☺



Random Data



Zero Data

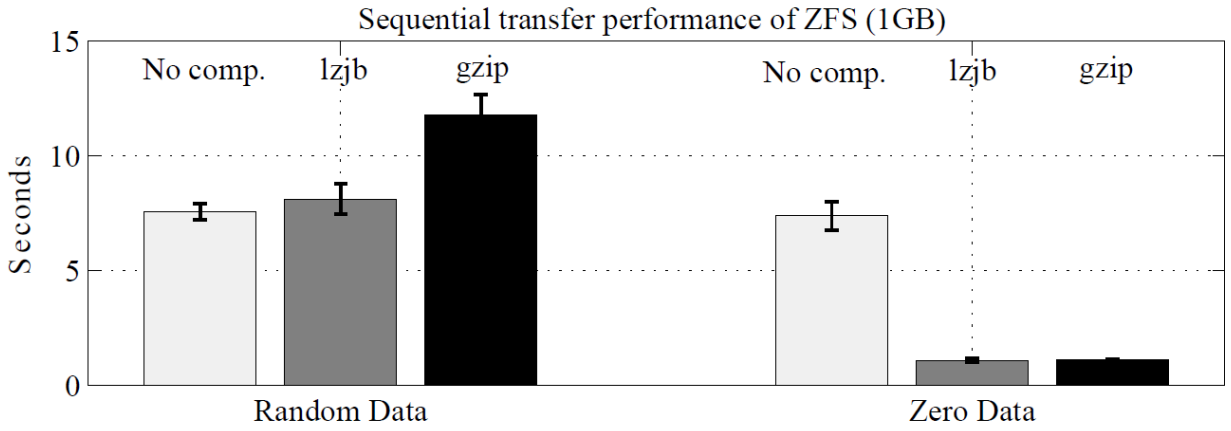
Introduction

- Benchmarking is essential to evaluate storage systems:
  - File systems, databases, micro-benchmarks…
  - FileBench, LinkBench, Bonnie++, YCSB,…
- Many storage benchmarks try to recreate real workloads:
  - Operations per unit of time, R/W behavior,…
- But, what about the **data generated** during a benchmark?
  - Real dataset: representative, proprietary, potentially large
  - Simple synthetic data (zeros, random data): not-representative, easy to create, reproducible

The Problem

- Does the benchmarking data actually matter?

- ZFS Example: A file system with built-in compression
  - ZFS is significantly content-sensitive if compression enabled
  - The throughput also varies depending on the compressor



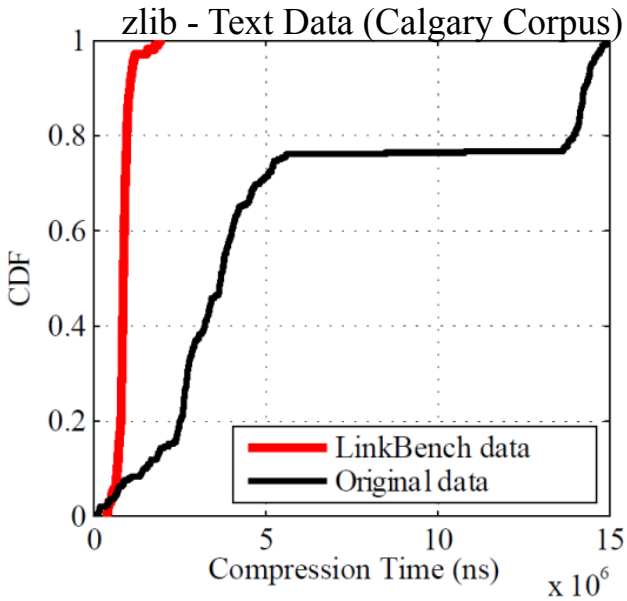
- **Conclusion**: Yes, it matters if data reduction is involved!

Current Solutions

- Some benchmarks try to emulate the **compressibility of data** (*LinkBench*, *Fio*, *VDBench*): *Mixing compressible/incompressible data at right proportion.*

50% compressible!

- **Problems** (*LinkBench* data vs real data):
  - Accurate compression ratios but insensitive to compressor
  - Unrealistic compression times
  - Heterogeneity is not captured



Our Mission

- Complex situation:
  - Most storage benchmarks generate unrealistic contents
  - Representative data is normally not shared due to privacy issues
- Not good for the performance evaluation of storage systems with data reduction techniques built-in.
- We need a common approach to **generate realistic and reproducible** benchmarking data.
  - In this work, we focus on compression benchmarking.

Summary of our Work

- **Synthetic Data GENerator (*SDGen*)**: *open and extensible framework to generate realistic data for storage benchmarks.*
  - **Goal: "mimic" real datasets.**
  - **Compact, reusable and anonymized dataset representation.**

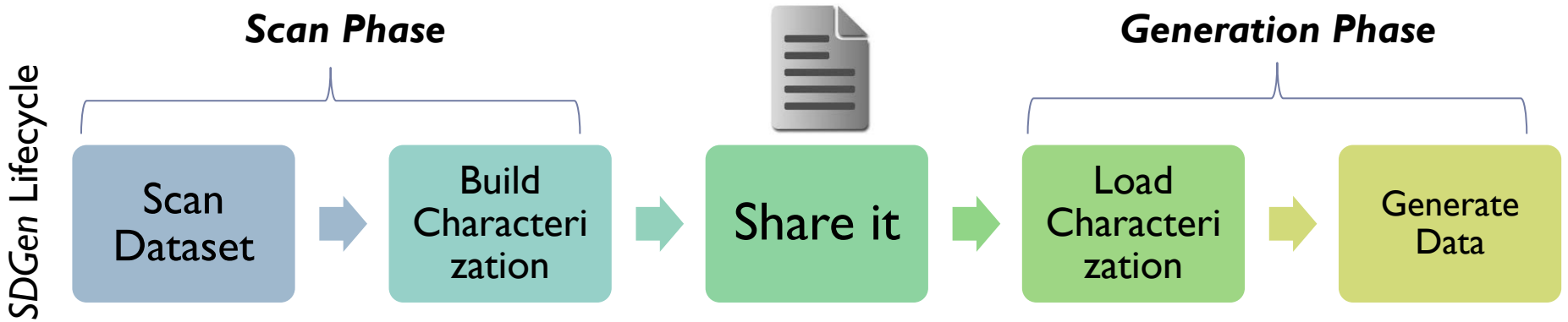
- **Mimicking compression**: identify the properties of data that are key to the performance of popular lossless compressors (e.g. zlib, lz4).

- **Usability and integration**: *SDGen* is available for download and has been integrated in popular benchmarks (*LinkBench*, *Impressions*).

SDGen: Concept & Overview

- **Mimicking method**: *capture the characteristics of data that affect data reduction techniques to generate similar synthetic data.*

- *SDGen* works in two main phases:



- *SDGen* can do *full scans* or *use sampling*.

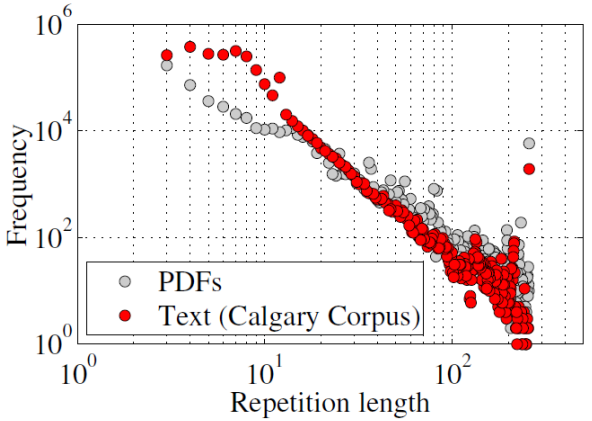
- *SDGen* requires knowing "**what to scan for**" and "**how to generate data**".

Mimicking data for compression

- We empirically found two properties that affect the behavior of compression engines:

- **Repetition length distribution**
  - **Key for compression time & ratio**
  - **Typically follows a power-law**

- **Byte frequency**
  - **Impacts on entropy coding**
  - **Changes importantly depending on data**



Generating synthetic data

- Goals:
  - Generate data with similar properties (repetition lengths, byte freq.)
  - Fast generation throughput

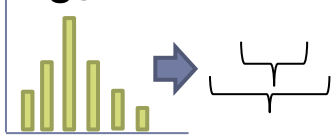
- At high-level, we generate a data chunk as follows:

1) Random decision: *repetition or not?*

2) Repetition: insert repeated data / Non-repetition: insert newly randomized data

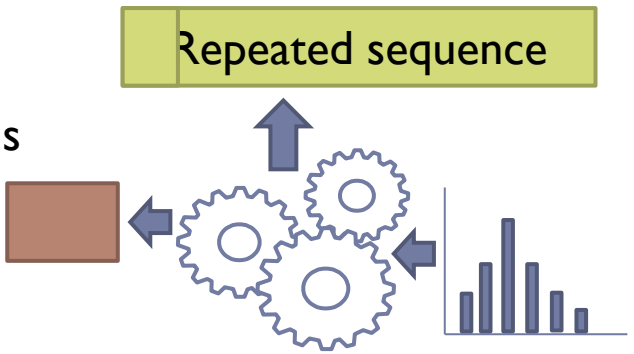
3) Pick a repetition sequence length from the repetition histogram

4) Insert repeated data



Rep. Len. Histogram

Initialize source of repetitions



Data generator

Byte Freq. Histogram

Synthetic chunk

Evaluation

Objective Metrics

- Compression ratio
- Compression time

Datasets

- Calgary/Canterbury corpus
- Silesia Corpus
- PDFs (FAST conferences)
- Media (IBM engineers)
- Sensor network data
- Enwiki9
- Private Mix (VMs, .xml, .html,…)

Additional Mimicked Properties

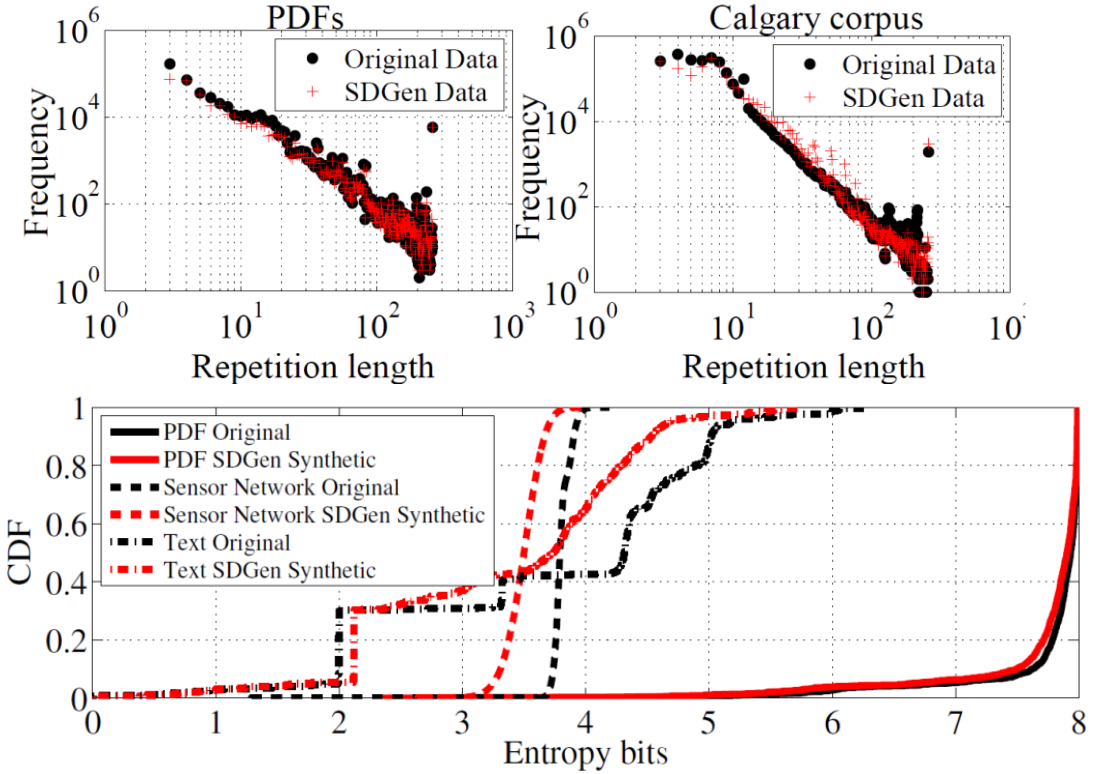
- Repetition length
- Entropy (byte frequencies)

Compressors

- **Target**: Lossless compression based on byte level repetition finding and/or on entropy encoding (*zlib*, *lz4*).
- We tested **other families** of compressors (*bzip2*, *lzma*).

Evaluation: Mimicked Properties

- **Experiment**: compare repetition length distributions and byte entropy in real and *SDGen* data.
- *SDGen* generates data that closely mimics these metrics.

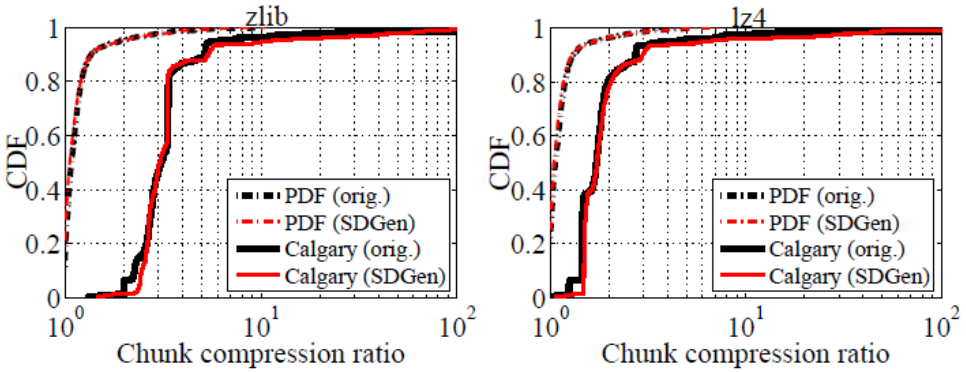


Evaluation: Compression Ratio & Time

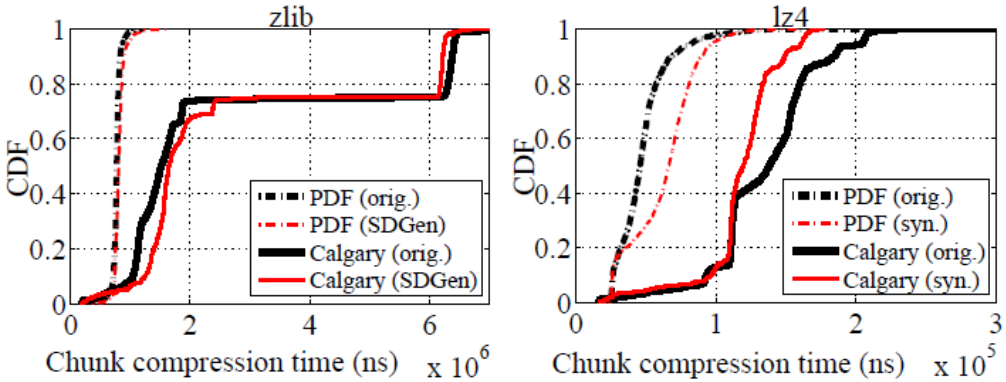
- **Experiment**: Capture per-chunk compression ratios and times for both synthetic and real datasets.

Per-chunk compression ratio

- Compression ratios are closely mimicked
- Heterogeneity is also well captured within a dataset



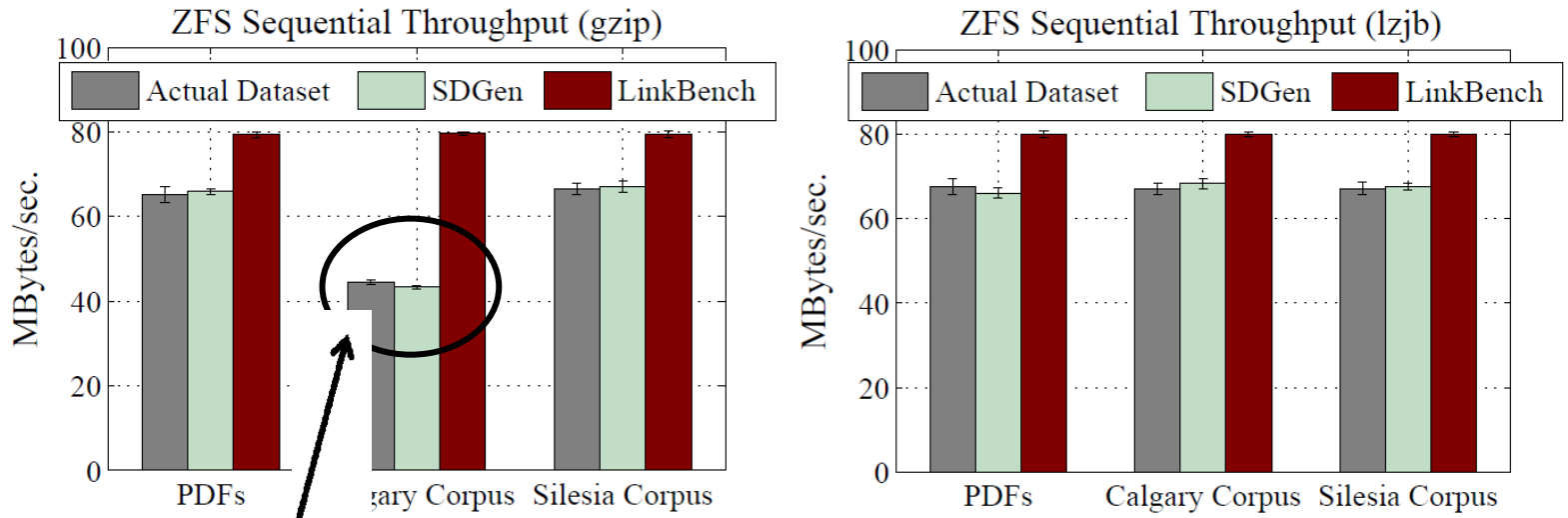
Per-chunk compression time



- Compression times are harder to mimic (especially for lz4)
- Still, for most data types compressors behave similarly

Evaluation: Performance of ZFS

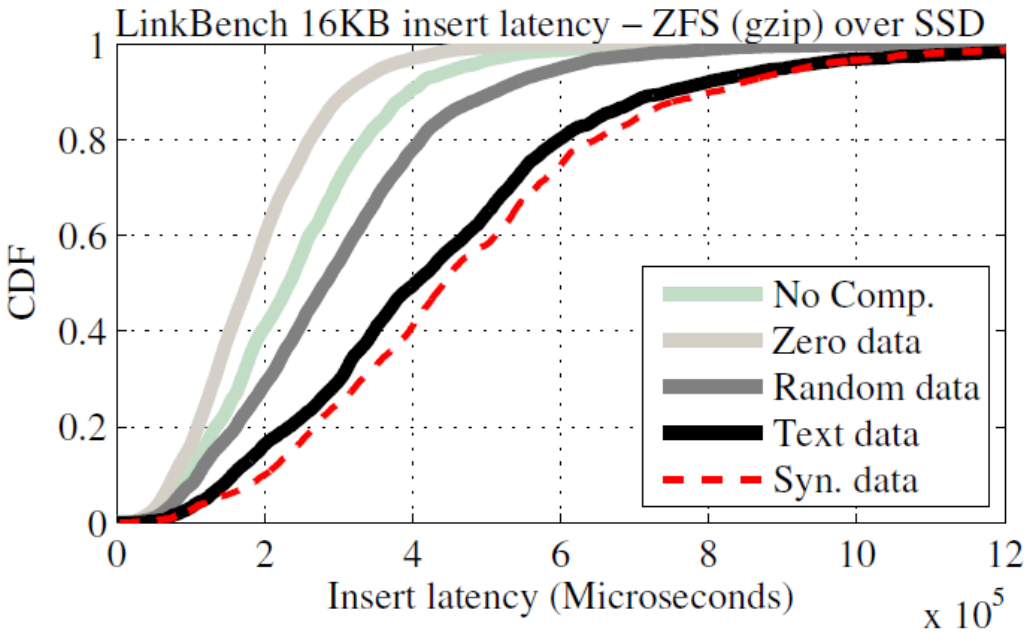
- **Experiment**: write to ZFS 1GB files augmenting previous datasets.
- ZFS exhibits similar behavior for both real and our synthetic data.
- ZFS digests faster *LinkBench* data (+12% to +44%).



DNA sequencing files in Calgary corpus are specially hard to compress

Evaluation: Integration with LinkBench

- **Experiment**: LinkBench write workload using distinct data types (ZFS + SSD storage).
- *SDGen* serves as data generation layer for *LinkBench*.
- Write latency is similar in both synthetic and text dataset.



Conclusions & Future Directions

- **Data** is an important aspect of storage **benchmarking** when **data reduction** is involved (compression, dedup).
- We presented *SDGen*: a **framework** for generating **realistic and sharable** benchmarking data.
  - **Idea**: scan data, build a characterization, share it, generate data
- We designed a method to mimic **data compression ratios and times** for popular lossless compressors.
- We plan to extend *SDGen* to mimic **data deduplication**.

Q&A

- Thanks for your attention!
- *SDGen* code: https://github.com/iostackproject/SDGen
- Funding projects:

Towards the next generation of open Personal Clouds

http://cloudspaces.eu

Software-Defined Storage for Big Data

http://iostack.eu

Backup: Generation Performance

- Characterizations of chunks can be used in parallel for generation.
- Generating uncompressible data is more expensive.
  - We plan optimizations to wisely reuse random data for boosting throughput.