Estimating Deduplication Ratios in Large Data Sets

Danny Harnik, Oded Margalit, Dalit Naor, Dmitry Sotnikov Gil Vernik

Estimating dedupe and compression ratios – some motivation

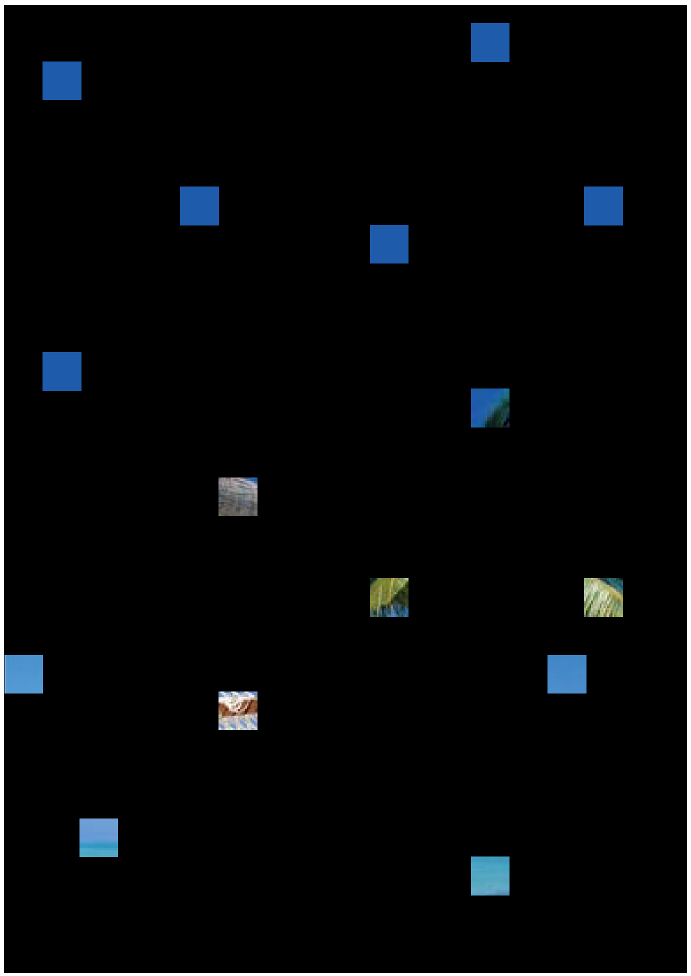
- Data reduction does not come for free
  - Incurs overheads, sometimes significant
  - Not always worth the effort
  - Better to know in advance

- Different techniques give different ratios
  - What technique to use?
    - Chunks? Fixed? Variable-sized? Full-file? Compression?
  - Sometimes better to consolidate storage pools, sometimes not

- Different data reduction ratio: different number of disks to buy
  - Disks = money !

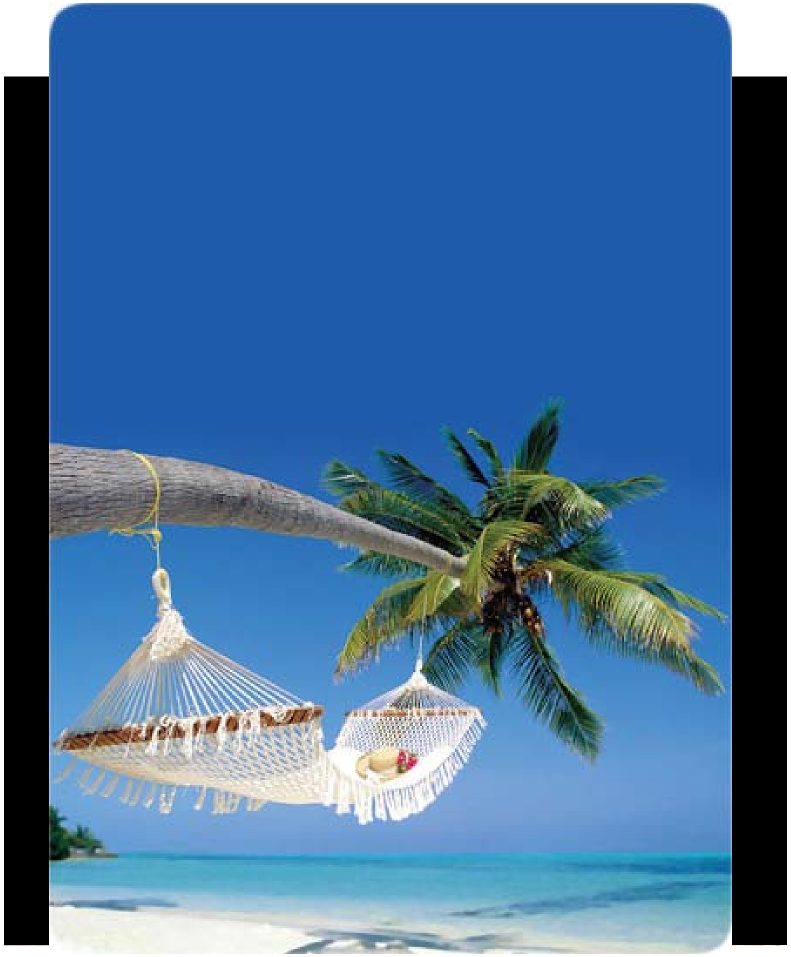
How do you estimate efficiently?

- **Option 1:** according to data type, application type, etc…
  - Can be grossly in-accurate
  - “I’ve seen the same DB application with dedupe ratio 1:2 and 1:50”
  - Better to actually look at the data.
- **Option 2:** Sample a small portion of the data set and deduce from it
  - Problematic when deduplication is involved



- Data set == picture
  - Identical block == identical picture block
- In real life we don't see the full data set at once
  - Rather, we probe locations
  - In sampling we only probe a small number of locations

Why sampling is problematic for dedupe estimation



- Dedupe ratio is 1:2
- In order to see the duplicates must hit the exact same location twice…
- Birthday paradox – need ~$N^{½}$ elements to hit a collision
  - Ω($N^{½}$) to see a large number of collisions.
- What about triple collisions?

Formal limitations of sampling for Distinct Elements

Lower bounds: Can show 2 data sets with far apart dedupe ratio, but same "behavior" under sampling of $O(N^{1-\varepsilon})$ elements

- Charikar, Chaudhuri, Motwani, & Narasayya 2000 - showed basic bounds and empirical failures of approximating "distinct elements".
- Rashkodnikova, Ron, Shpilka & Smith 2009 – need $\Omega(N^{1-\varepsilon})$ samples to approximate to within a multiplicative factor.
- Valiant & Valiant 2011 - need $\Omega(N / \log N)$ samples for same task.

Bottom line: need to look at essentially he whole data set

Algorithms that see the whole data set : challenges

- Still a hard task
- High resources requirements: memory, time, disk accesses

- Naïve approach: simulate the actual dedupe
    - just don't store the data
  - Problem: simply storing the table of hash indices is too big for memory
    - E.g. [Meyer & Bolosky, FAST 11]
- Example: metadata for 7 TB of data → 24 GB of metadata
  - Using an efficient dynamic data structure will require additional overhead

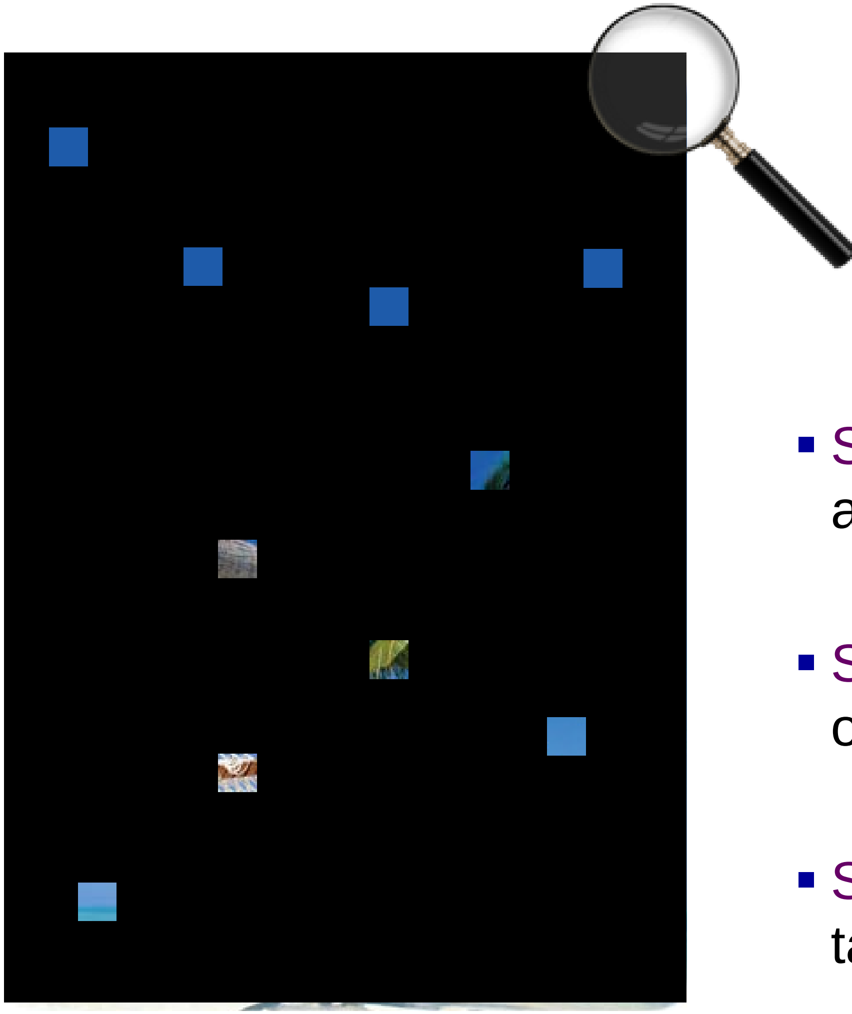
- Many solutions, none easy

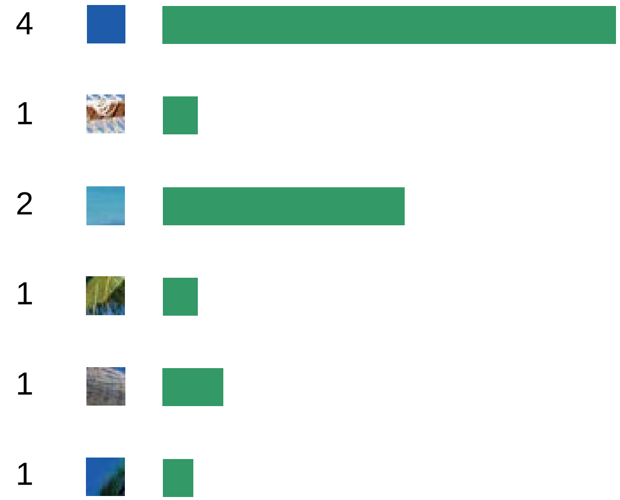
- Goal 1: find a memory efficient estimation scheme with high accuracy
- Goal 2: Be as efficient as possible (time & CPU).
  - More pronounced when need to integrate local (LZ) compression with deduplication

Plan for rest of talk

- Our Algorithm
- Integrating compression
- Analysis – both formal and empirical
- Related work
- Full-file deduplication – a special case

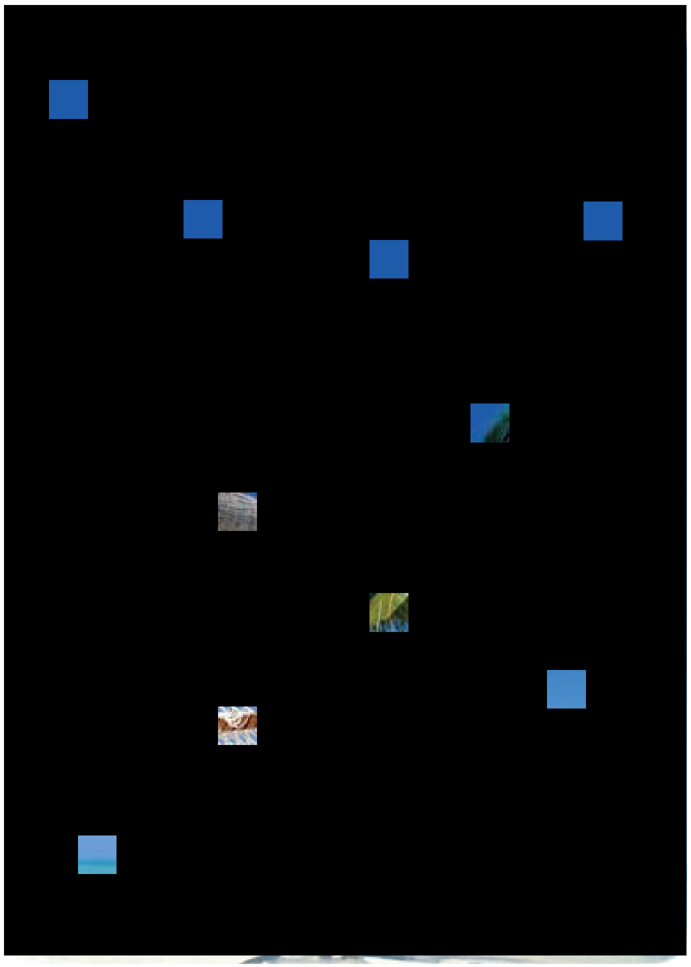
Our Algorithm: Sample & Scan

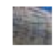




- Sample phase – create a "base sample" and store in memory
- Scan phase – Count appearances only of elements in the base sample
- Summary – Average of the base sample tallies too give estimation

What about compression?



4		0.04
1		0.9
2		0.05
1		0.6
1		0.2
1		0.4

- Add compression evaluation only to elements in the base sample
  - Only in the sample phase
  - Crucial since compression is a time consuming operation
- In the summary take the compression stats into account

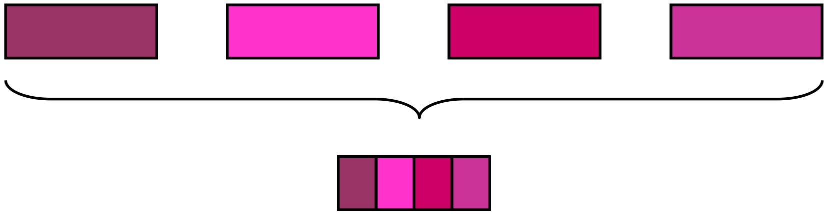
- Note: estimating dedupe and compression ratios separately is not sufficient for accurate estimation of the combination

Formal Analysis

- We prove a relation between the size of the base sample and the accuracy achieved for a given confidence parameter
  - Accuracy – by how much we may error
  - Confidence – with what probability

Intuition:

- Each block in the data-set has a "contribution" to the overall reduced output
  - Example: Suppose a block's dedupe count is 4
    - then each block contributes ¼ of a block to the overall output



- We attempt to estimate the average contribution over the whole data set
  - Average contribution = total output size / total input size
- Estimating of averages is well studied
  - Our estimation should form a normal distribution around the actual value

Example of Size of Sample vs Error and Confidence

Dedupe Ratio	*Error*	*Confidence*	*Sample Size*	*Memory*
3:1	***0.01***	***0.0001***	***44557***	***10.7 MB***
5:1	***0.01***	***0.0001***	***1237938***	***29.7 MB***
15:1	***0.01***	***0.0001***	***11142000***	***267 MB***

- Error is a percentage of the output size
  - Good dedupe → small output → 1% error is much smaller in absolute values
  - This is why some bound on dedupe ratio is required
- If error percentage of original size is acceptable, then life is easier
  - and memory requirements become even smaller

Empirical Analysis

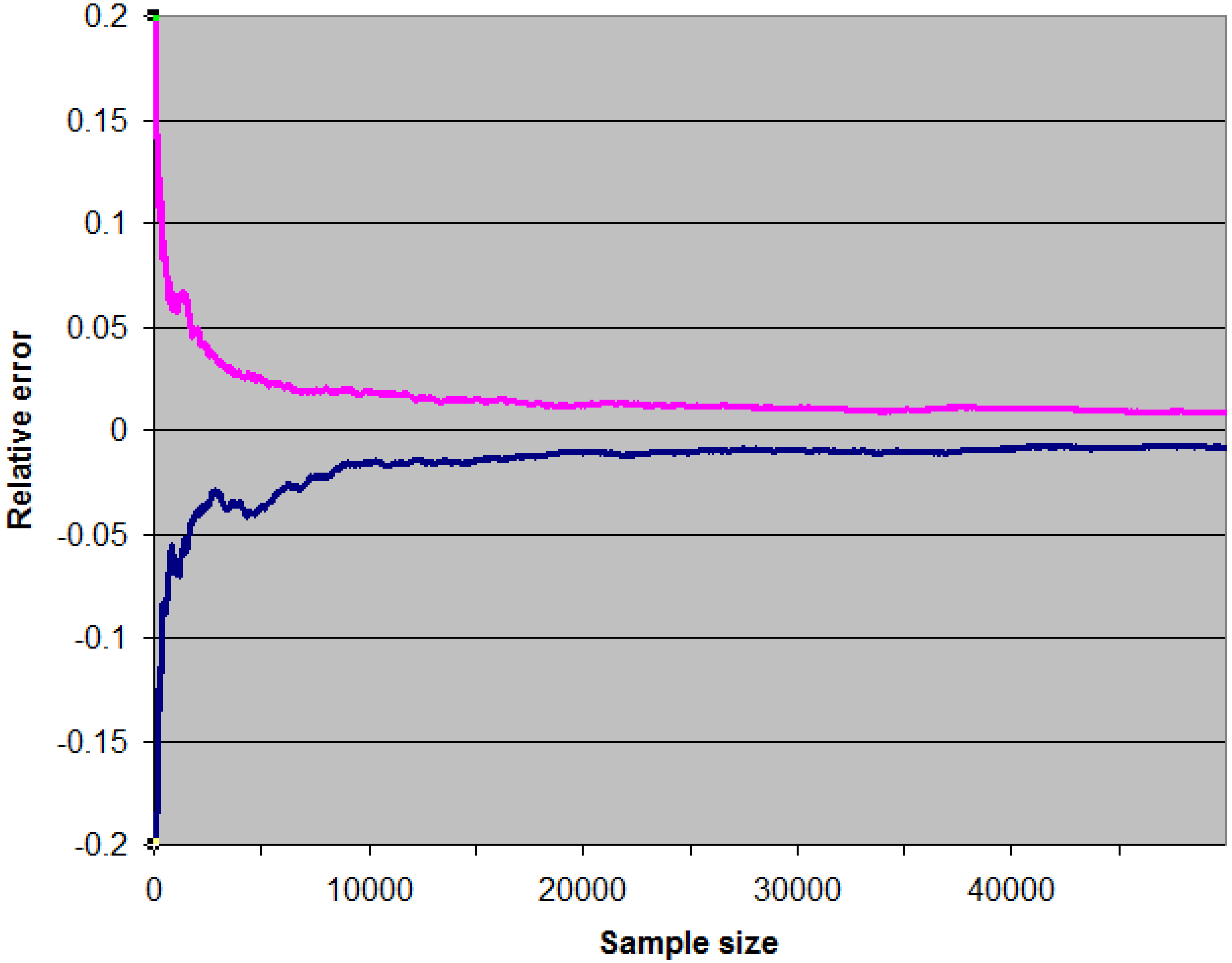
- Evaluated our estimation on 4 real life data-sets:

1. **Workstation data**
2. **File repository of an organization**
3. **Backup repository**
4. **Exchange DB periodical backups**

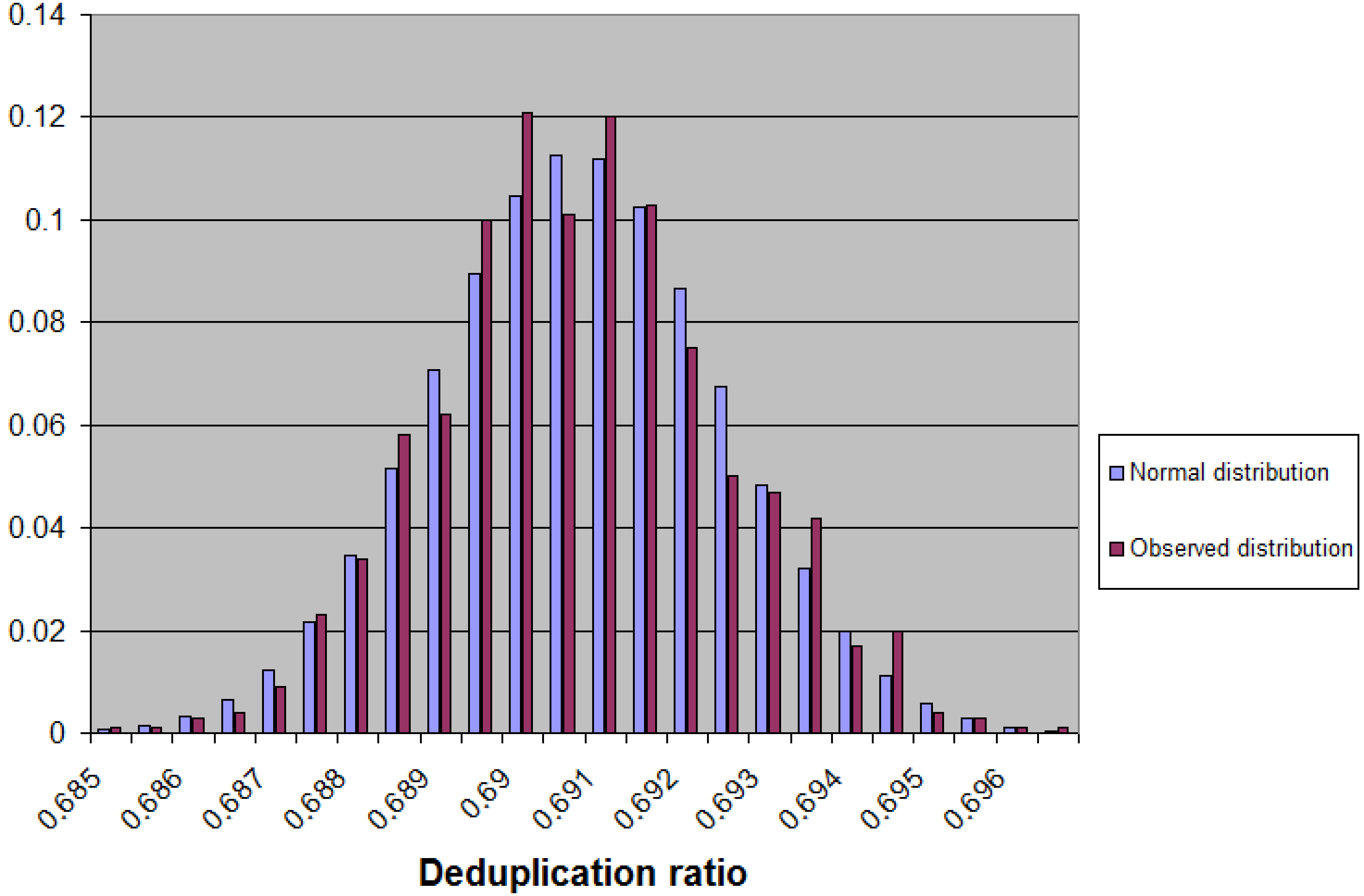
- Largest data set was the file repository with 7TBs of data

- The tests back-up our formal analysis

Convergence of the error percentage as base sample grows



Error is indeed normally distributed around the actual ratio



Related Work

- Distinct elements is a heavily studied topic
  - DB analysis
  - Streaming algorithms
- [FM85],…,[AMS99],…,[KNW2010],…many more…

Distinct elemnts works:

- Focus on one-pass algorithms
- Always fixed size elements – not files or variable sized blocks
- Do not consider compression

Our work:

- not one pass
- Comparable to most works (memory wise)
- When combined with compression we give the best performance (minimal number of chunks to compress)

Full-file deduplication

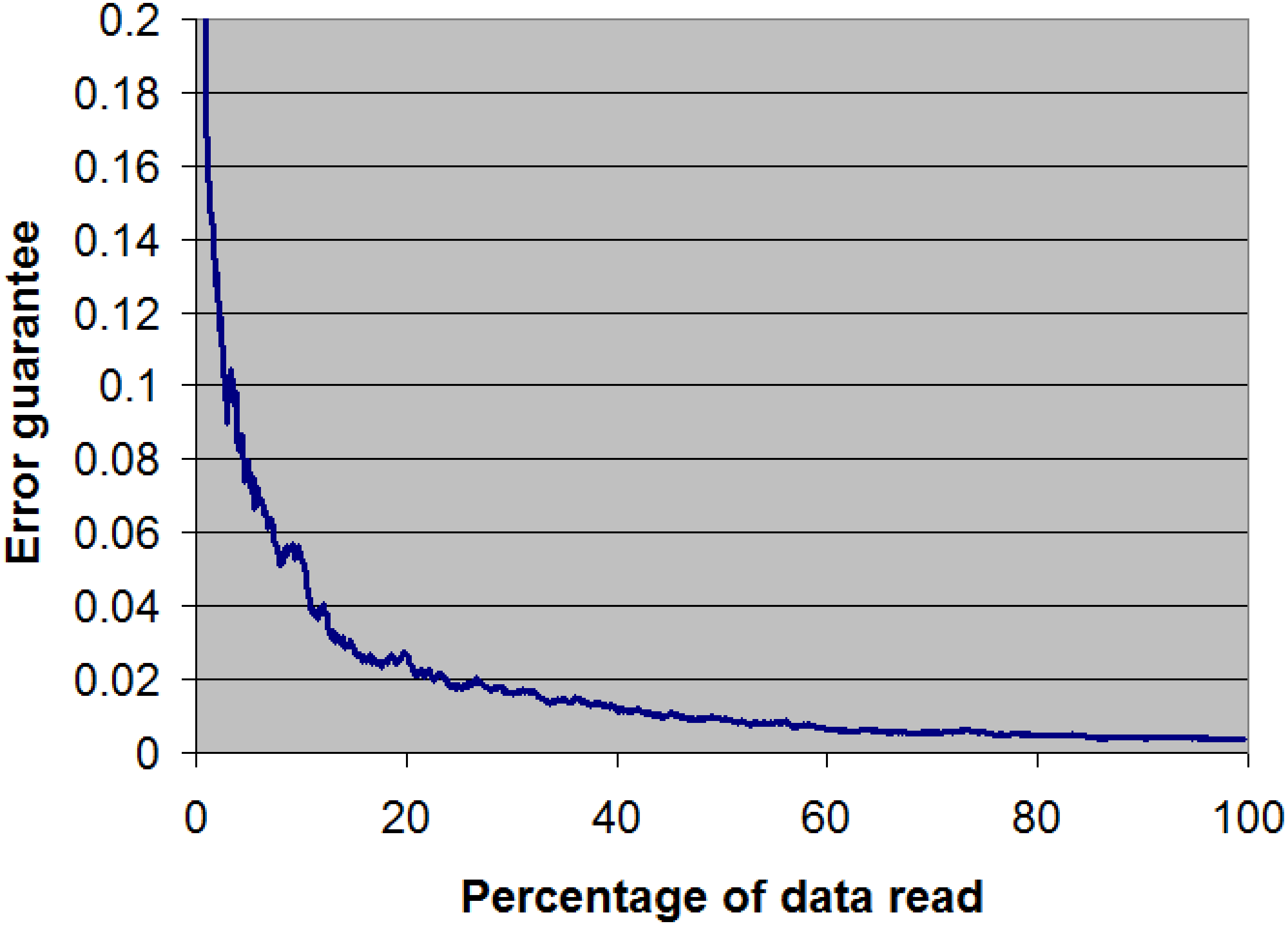
- Files have properties and metadata that can help us
  - Different file length → no dedupe
  - Different hash on first block → no dedupe

Our algorithm:

- In sample phase keep also the file length and first block hash of base sample
- During scan phase:
  - Check if file size relevant to base sample, if not, discard
  - Check if first block hash relevant to base sample, if not, discard
  - If still relevant, then read whole file from disk

Overall: all metadata is scanned, not all files read!

Full-file estimation:
Percentage of data read from disk vs. desired error gurantee



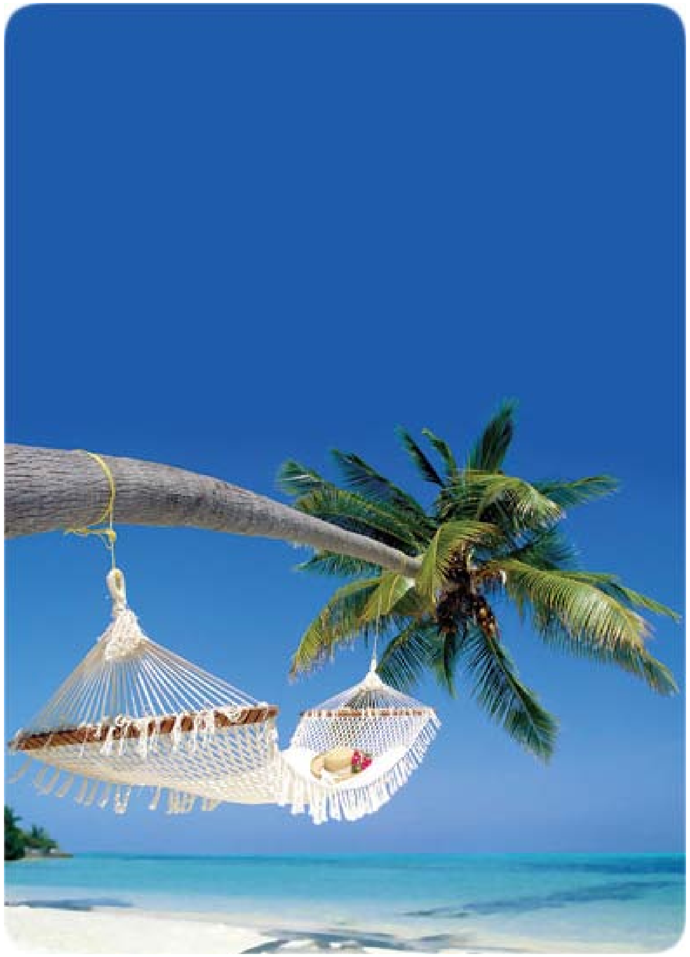
What to take home from this talk?

1. Need to be careful when doing sampling for dedupe

2. We have a good algorithm if you can run a full scan or already have metadata available.

3. Our algorithm integrates compression and dedupe naturally
   - Practically no overhead when adding compression

4. For full-file deduplication we reduce the data reads substantially!



Thank You !