IBM Research labs - Haifa

Estimating Unseen Deduplication – from Theory to Practice

Danny Harnik, Ety Khaitzin, Dmitry Sotnikov



Deduplication and Estimation

- Deduplication = removing duplicates in storage

- **Deduplication Estimation**: Given large a large dataset, understand the potential benefit from deduplication.

- **Why estimate?**
  - **Sizing:** Different data reduction ratio ⇨ different amount of storage to buy ($$$!)
  - **Data placement**
  - **Choosing storage**
  - **Customers are asking for help…**

- Rule of thumb estimations exist, but:
  - The savings for similar data types can vary greatly between users
  - To really know you must look at the actual data.

Deduplication Estimation

- **Obvious solution:** scan the entire dataset
  - Extremely taxing for very large datasets

- Want to estimate without seeing all of the data!
  - Easy for compression [HKMST13]
  - Hard for deduplication [HMNSV12]
    - Hardness stems from the global nature of deduplication
- **Bottom line:**
  - Need to sample a very large fraction in order to be accurate.
  - [VV11] – need $\Omega(N / \log N)$ samples

Talk plan:

1. Solutions in practice
2. Solutions in Theory
3. From Theory to practice…

Note 1: Discussing deduplication in general

- Could be **Fixed size** or **variable sized** chunking
  - our tests use fixed 4KB chunks
- Estimate potential – perfect deduplication.

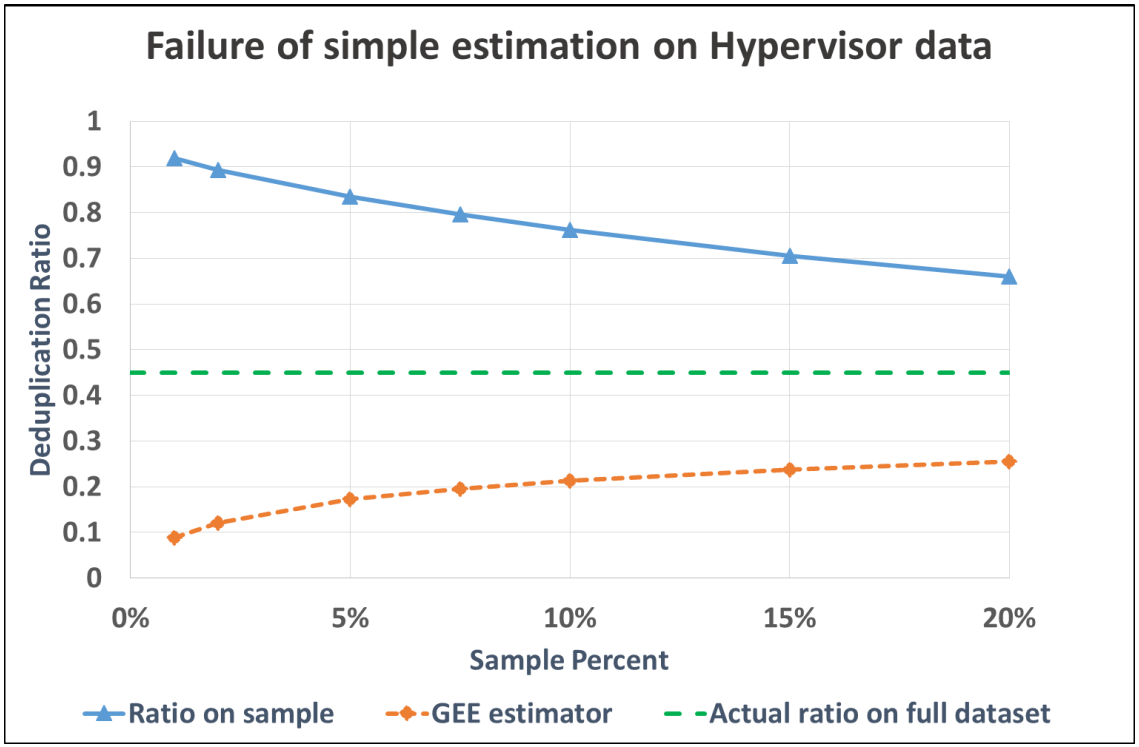
Note 2: Important related questions

- Distinct elements (data bases)
- How many different species? (Biology)

Solutions (in practice)

1. **Full scan:**
   - Low memory streaming algorithms
   - Deduplication specific solutions **[HMNSV12], [XCS13]**

2. **Sampling based:**
   - Heuristic approaches:
     - Ratio on sample
     - Statistical estimators
       - [S81]
       - [C84]
       - [HNSS95], [HS98]
       - [CCMN00]
       - …



Solutions (in theory)

- **“Estimating the Unseen”, Valiant & Valiant** [VV11], [VV13]
  - Algorithm with provable accuracy with a sample of size $O(N / \log N)$
  - Matches the lower bound

- **But… lots of problems moving from theory to practice**

Our results

- What sample size is enough?
  - Present a new method to gauge the accuracy of the estimation

- How to sample? No clear speedup in random reads…
  - Show how to get accuracy when sampling at a 1MB granularity
  - Present a new simple and stateless sampling algorithm

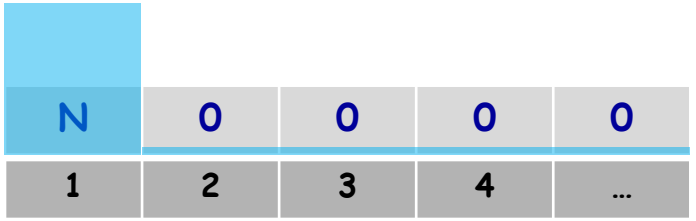
- High memory requirements from the algorithm
  - Propose 2 low memory variants of the algorithm

- What about compression?
  - Devise a combined deduplication and compression estimation method.

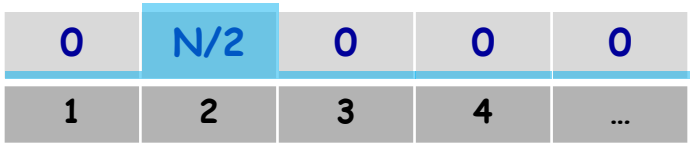
Duplication Frequency Histogram (DFH)

- The **DFH** x of a data set is a histogram $\{x_1, x_2, x_3, \ldots\}$
  - x_i how many chunks have duplication exactly i

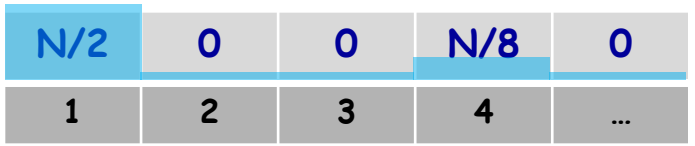
- Examples:
  - N unique values (no duplication):



  - All values appear exactly twice

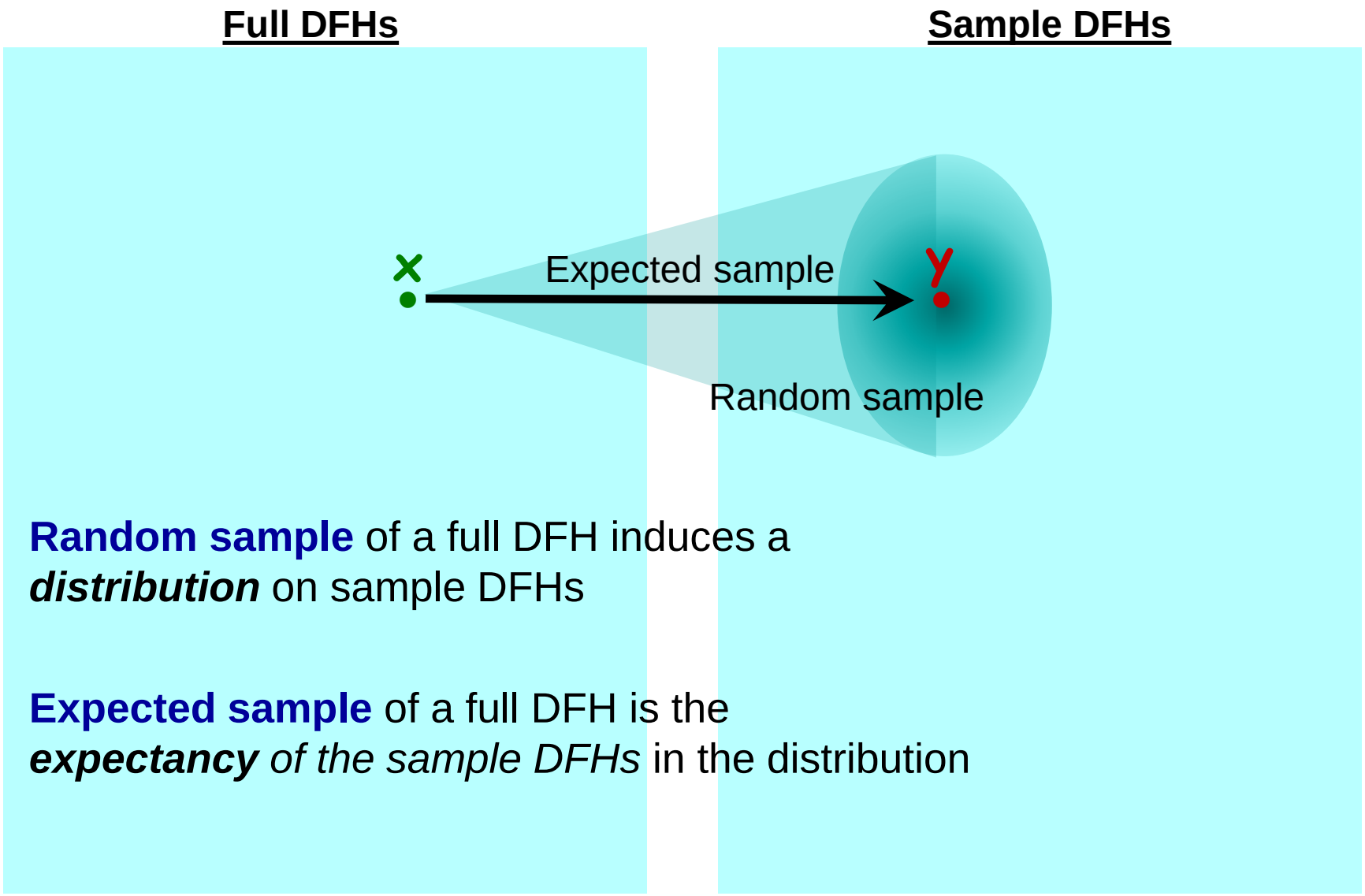


  - Half of the set are unique values and half appear 4 times

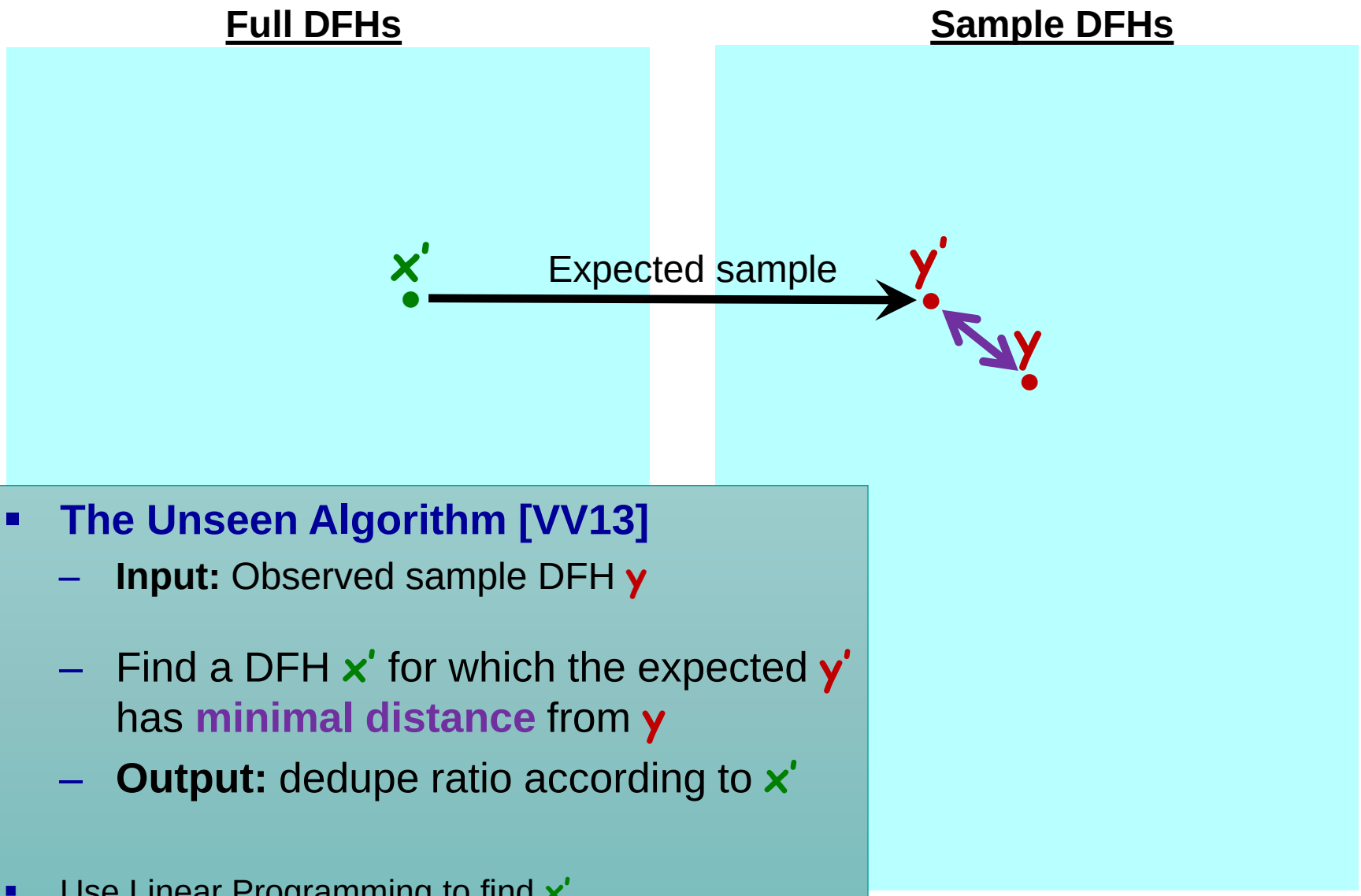


- **A small representation**
- **Fully characterizes the duplications in the data set**
- Can take a **DFH** of a full set or of a sample of the set

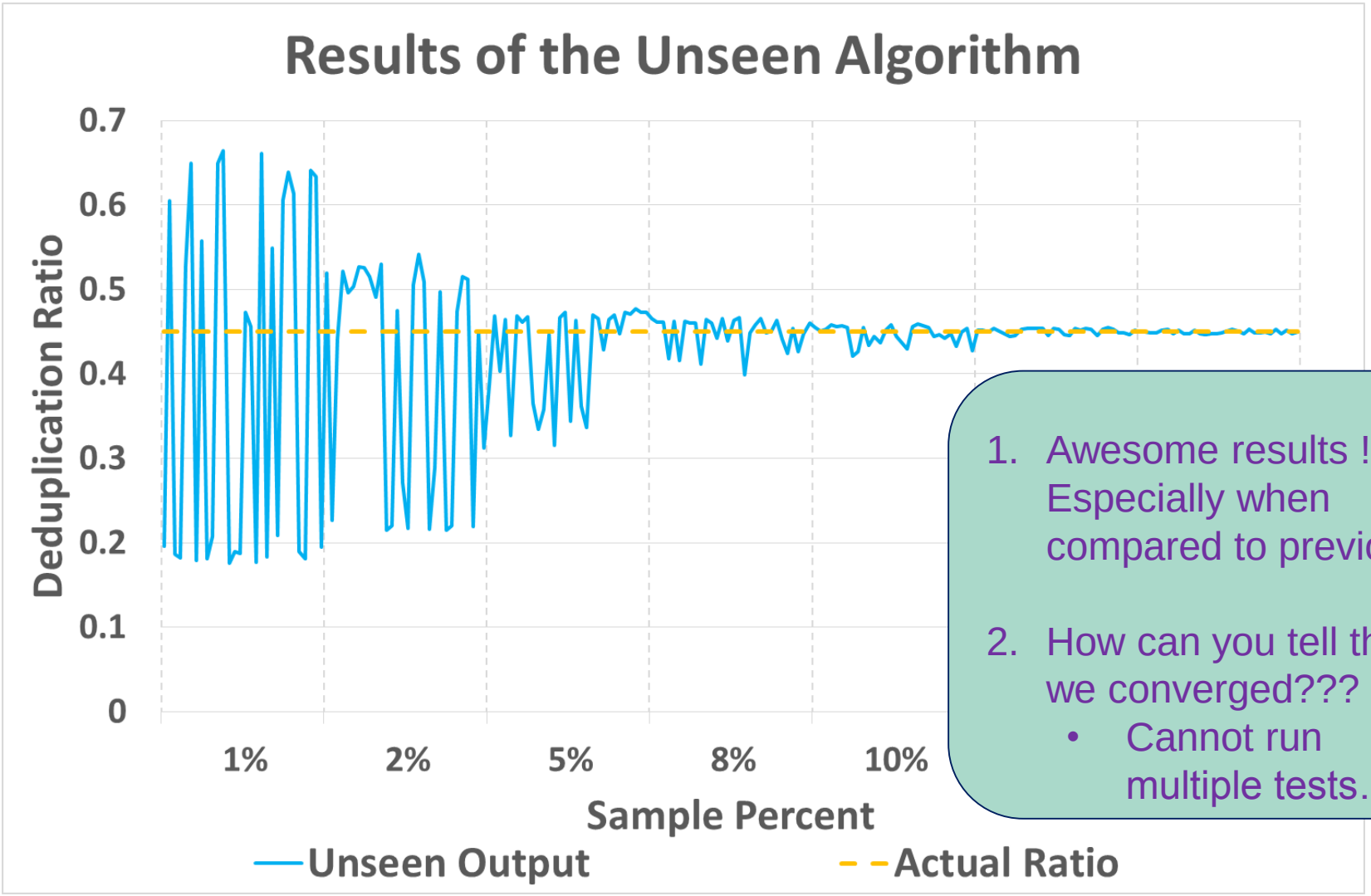
Map between full DFHs and sample DFHs



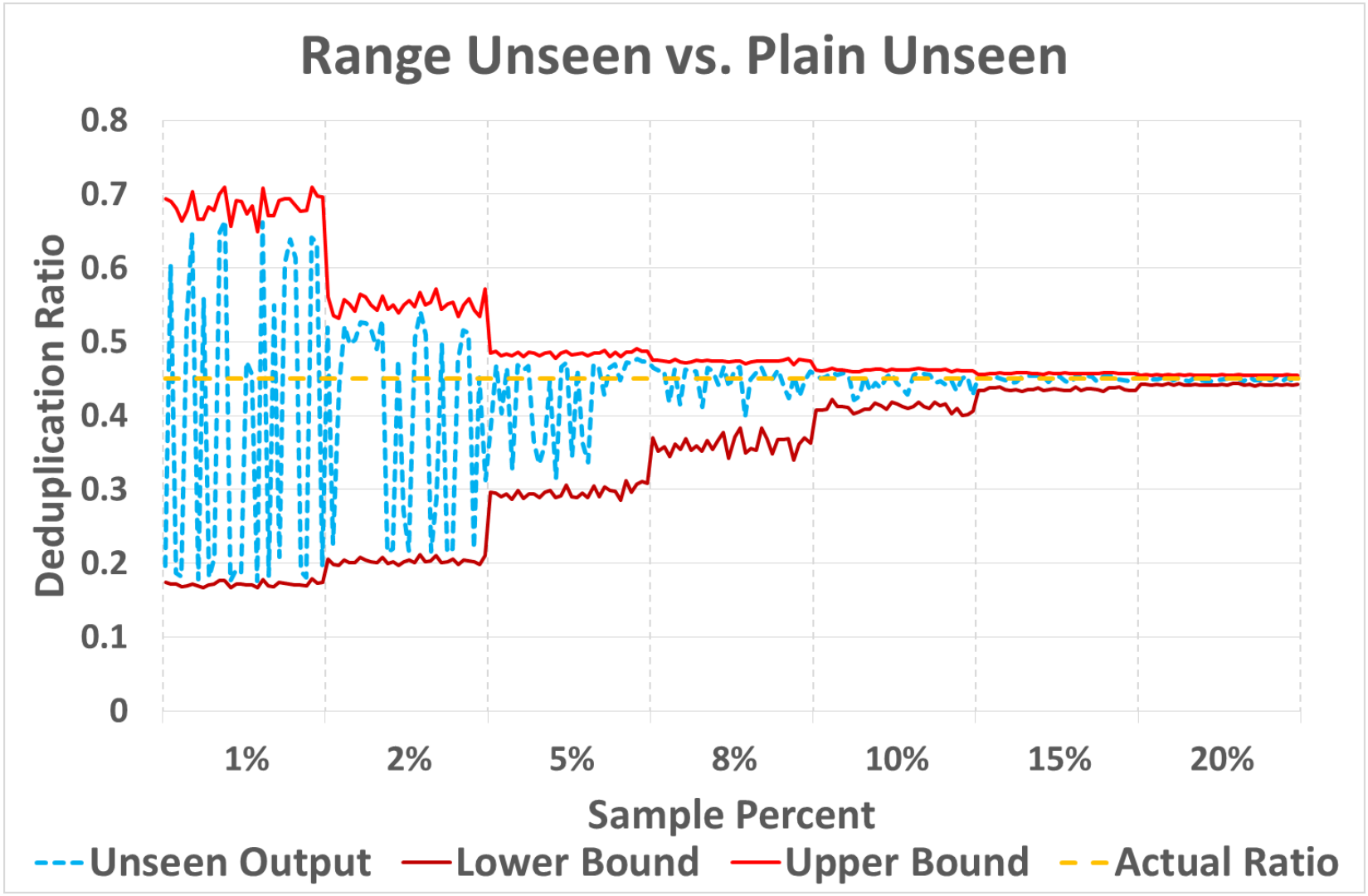
- **Random sample** of a full DFH induces a ***distribution*** on sample DFHs
- **Expected sample** of a full DFH is the ***expectancy*** *of the sample DFHs* in the distribution



- **The Unseen Algorithm [VV13]**
  - **Input:** Observed sample DFH **y**
  - Find a DFH **x'** for which the expected **y'** has **minimal distance** from **y**
  - **Output:** dedupe ratio according to **x'**
- Use Linear Programming to find **x'**

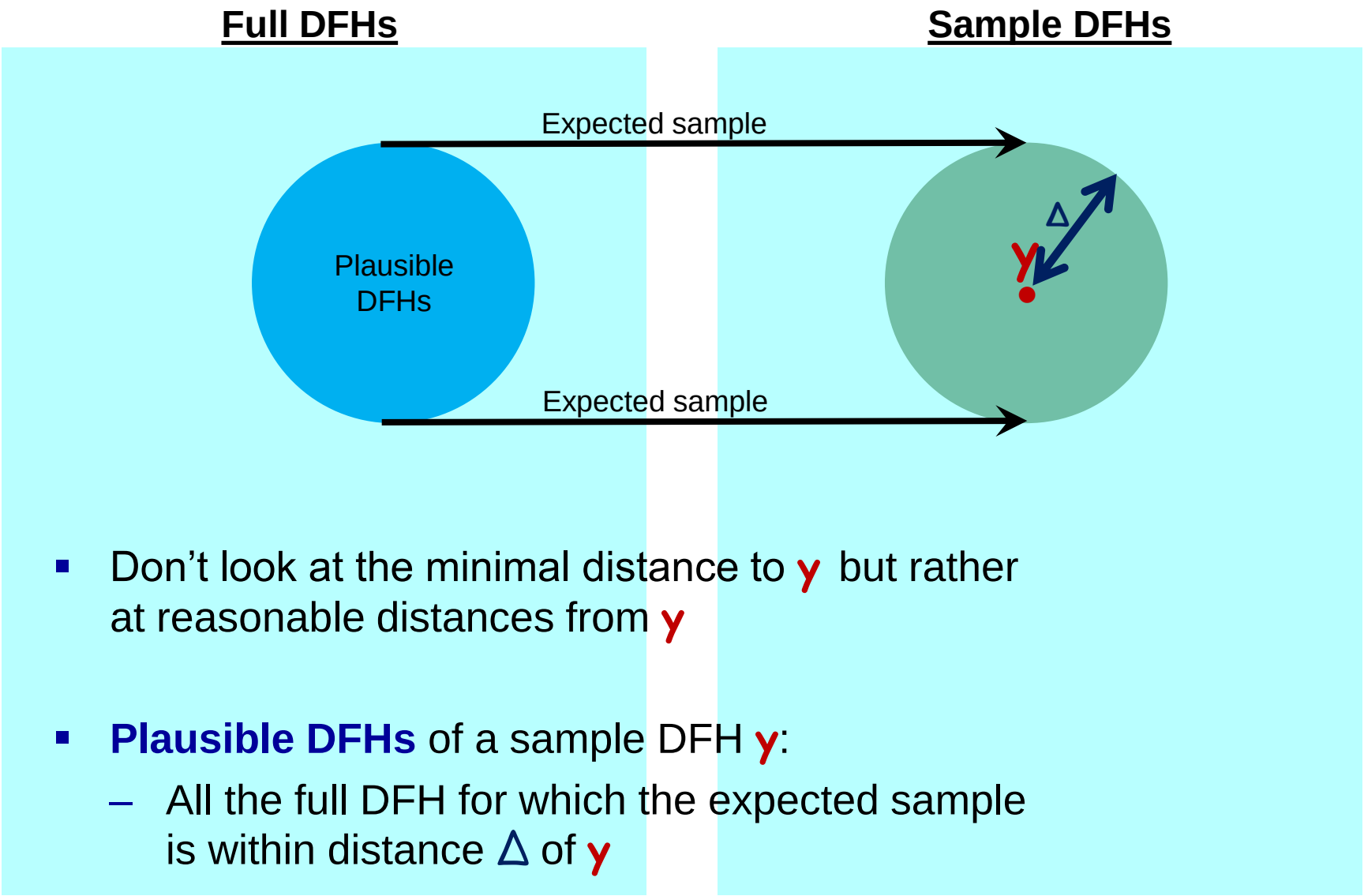


- Example of 30 independent tests at each sample size
- Same hypervisor data from previous slide



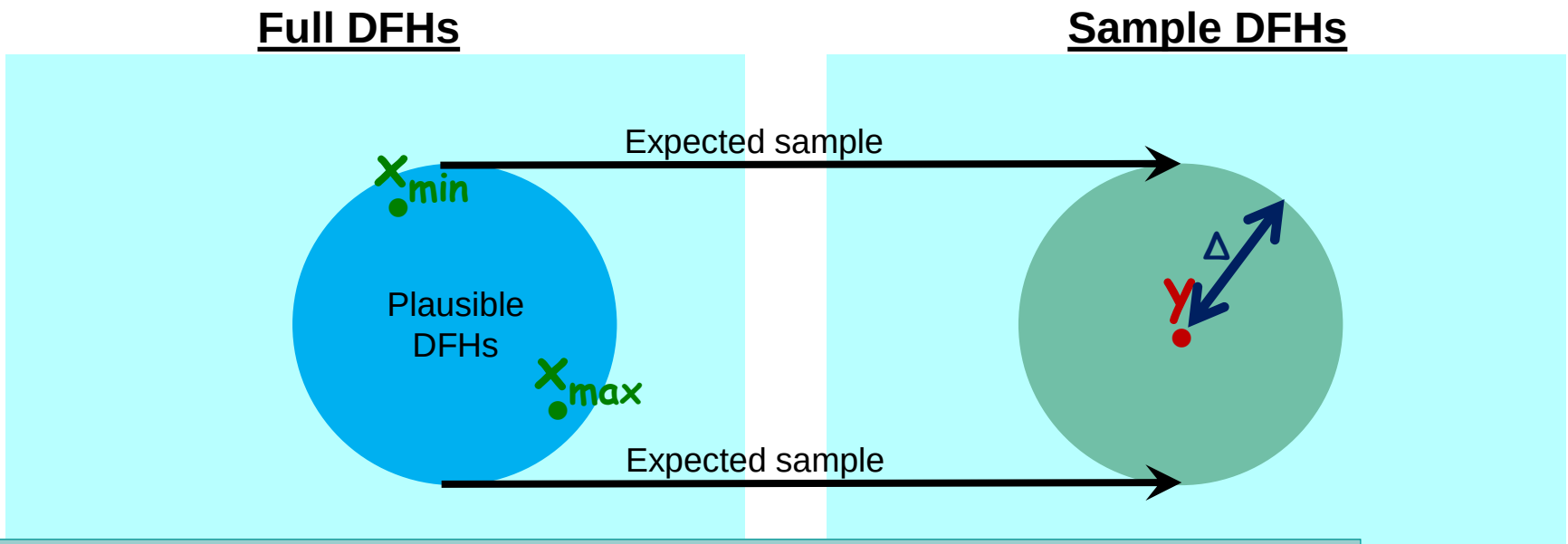
- **"Range Unseen"**
  - returns upper and lower bounds rather than estimation
  - W.h.p. the actual ratio lies inside the range

The Range Unseen algorithm - definitions



- Don’t look at the minimal distance to **y** but rather at reasonable distances from **y**
- **Plausible DFHs** of a sample DFH **y**:
  - All the full DFH for which the expected sample is within distance Δ of **y**

The Range Unseen algorithm



- **The Range Unseen Algorithm**
  - **Input:** Observed sample DFH **y**
  - Of all the plausible DFH of **y**
    - find the DFH x_{min} with the minimal dedupe ratio
    - Find the DFH x_{max} with the maximal dedupe ratio
  - **Output:** dedupe ratios of x_{min} and x_{max}
- Use Linear Programming to find x_{min} and x_{max}

Our results

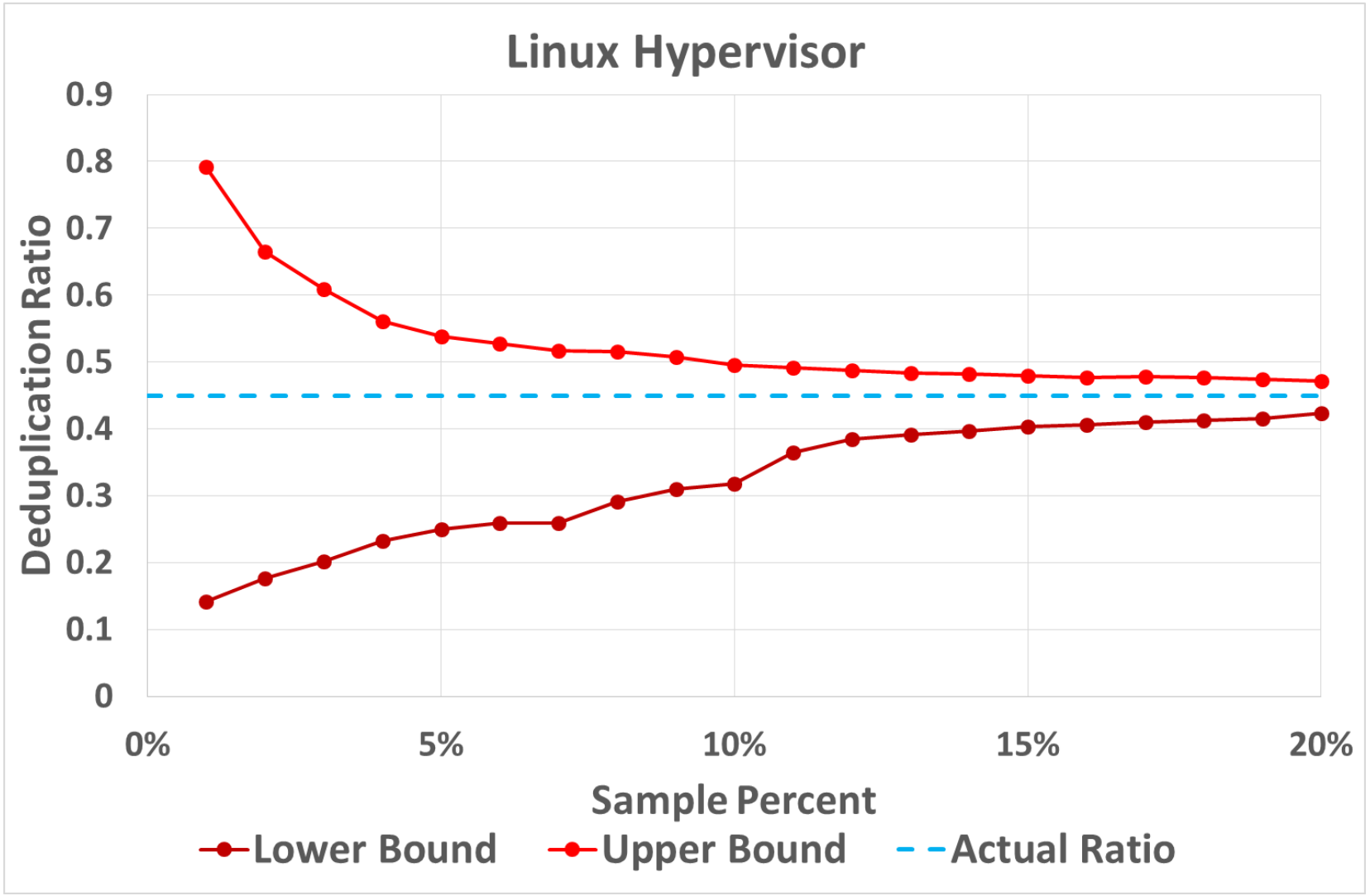
- What sample size is enough?
  - Present a new method to gauge the accuracy of the estimation

- How to sample? No clear speedup in random reads…
  - Show how to get accuracy when sampling at a 1MB granularity
  - Present a new simple and stateless sampling algorithm

- High memory requirements from the algorithm
  - Propose 2 low memory variants of the algorithm

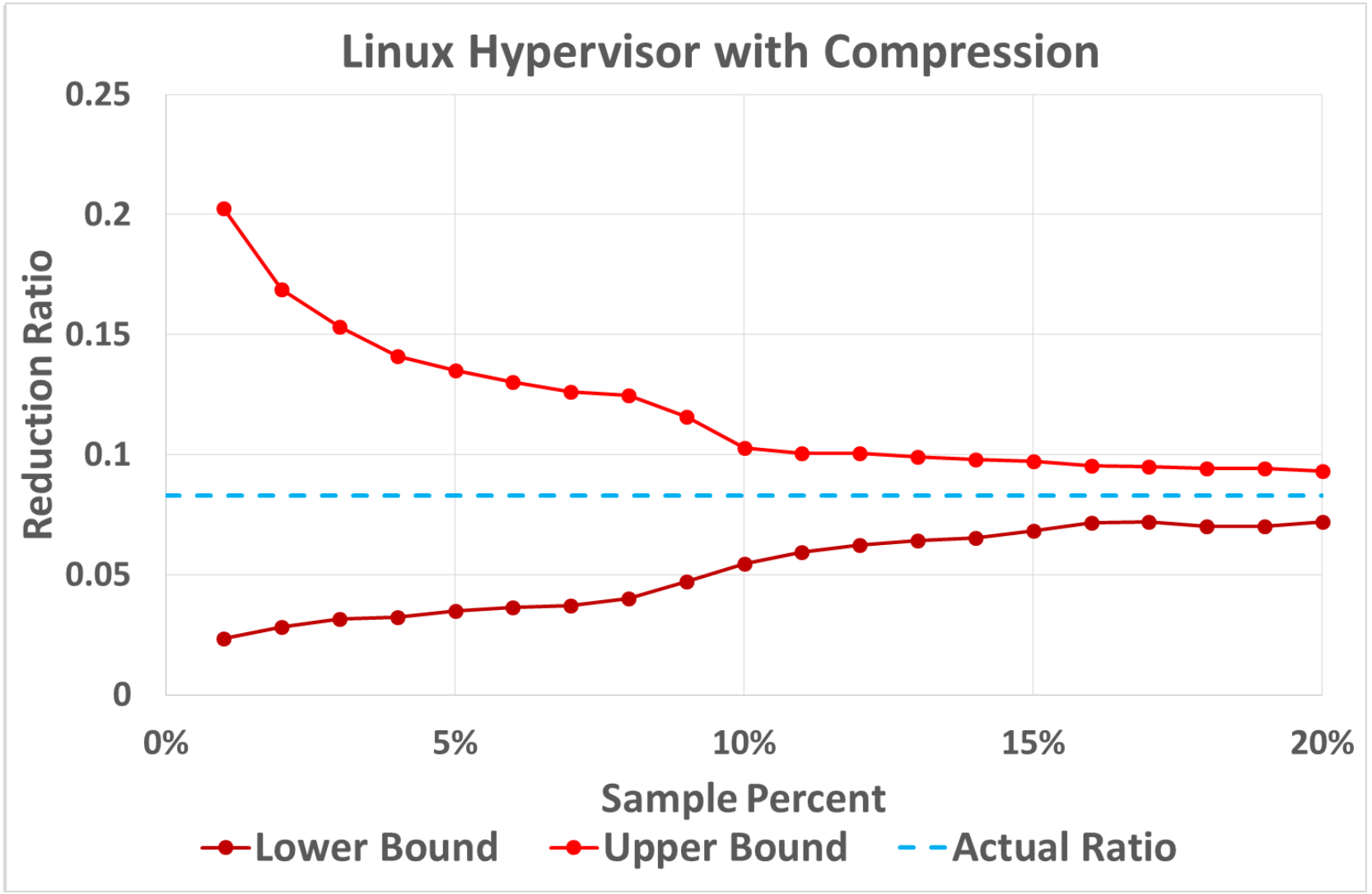
- What about compression?
  - Devise a combined deduplication and compression estimation method.

Putting it all together



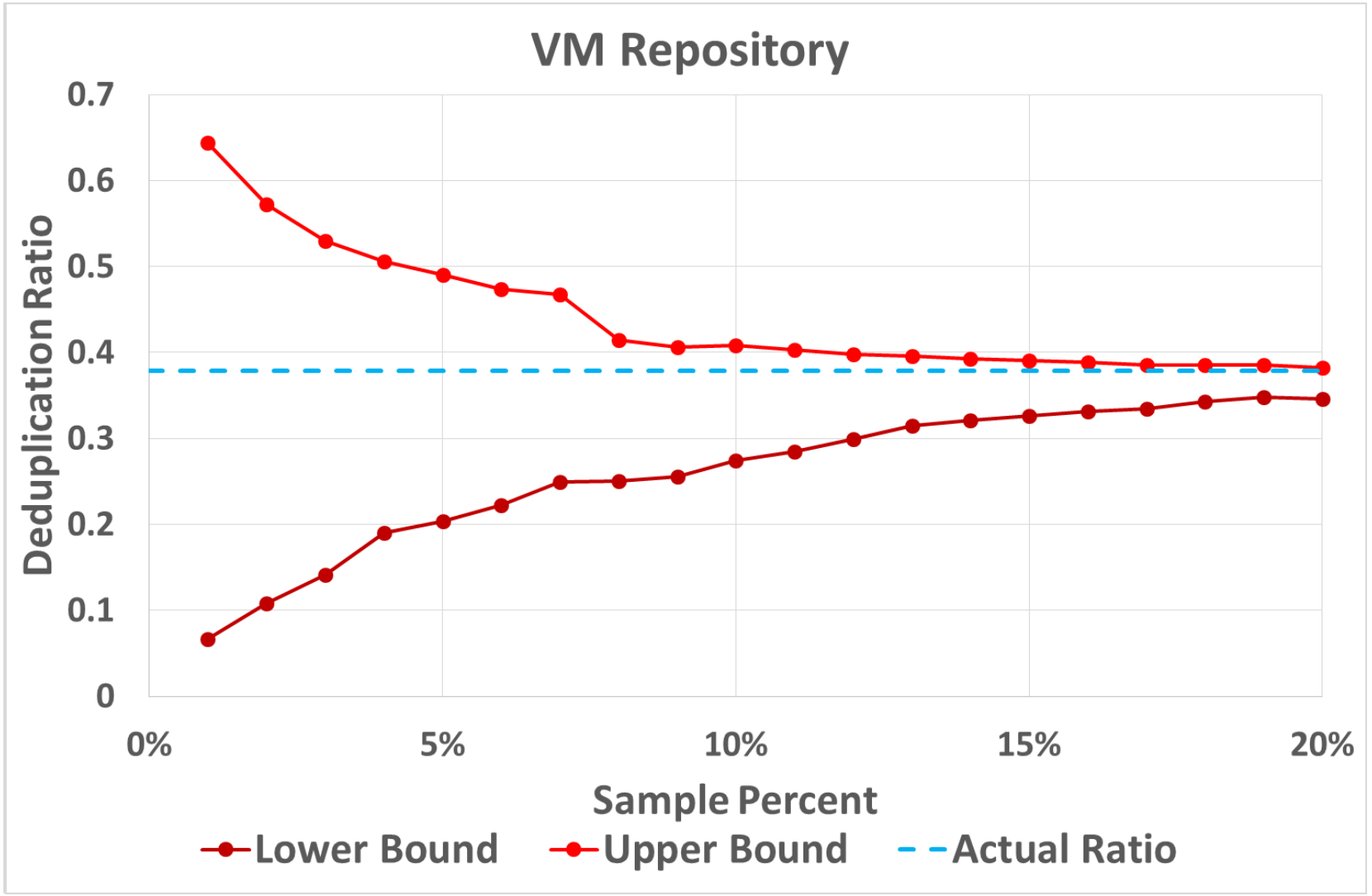
– includes low memory and reading at 1MB super-chunks

Putting it all together

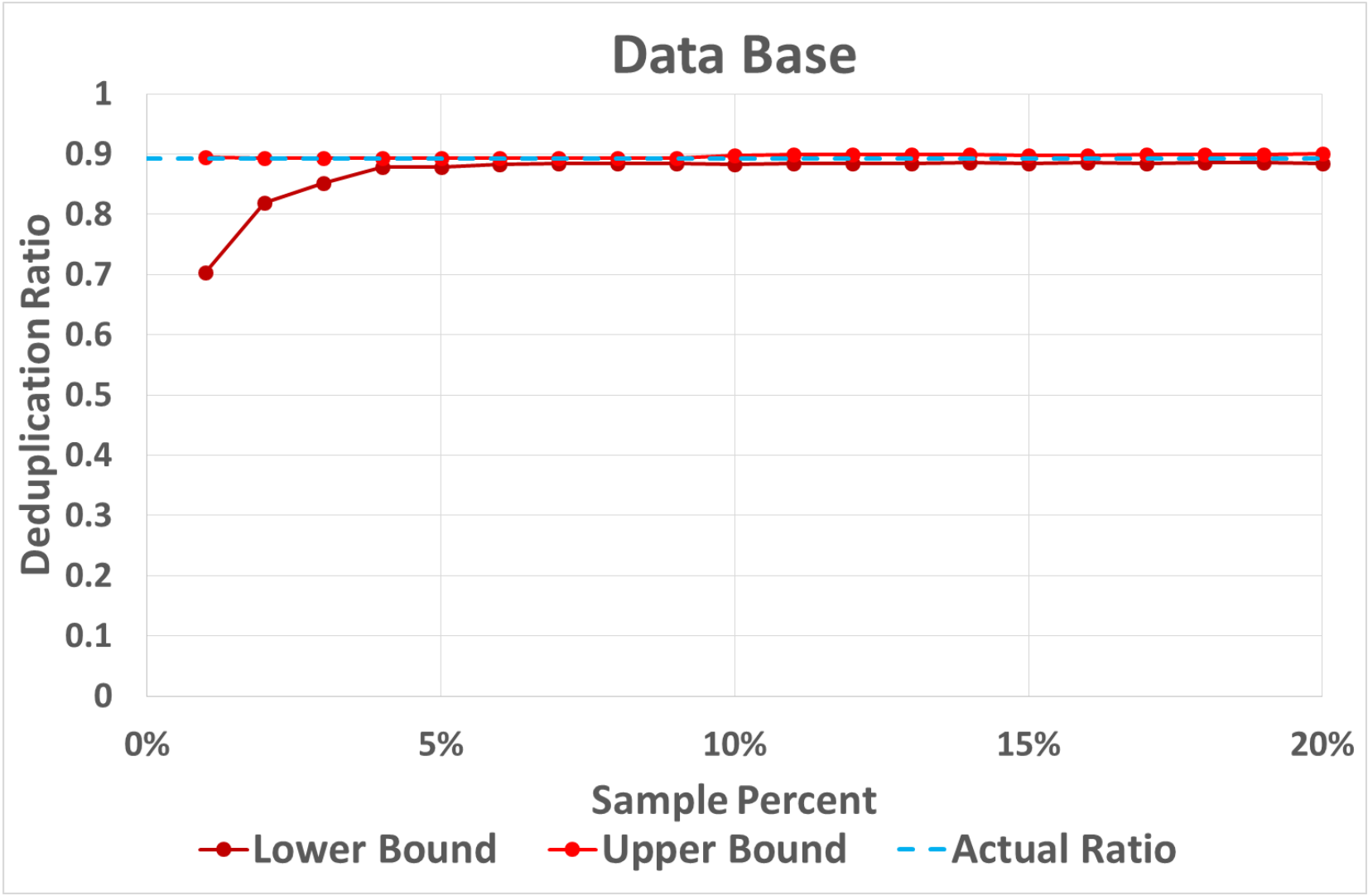


– includes low memory and reading at 1MB super-chunks

Putting it all together

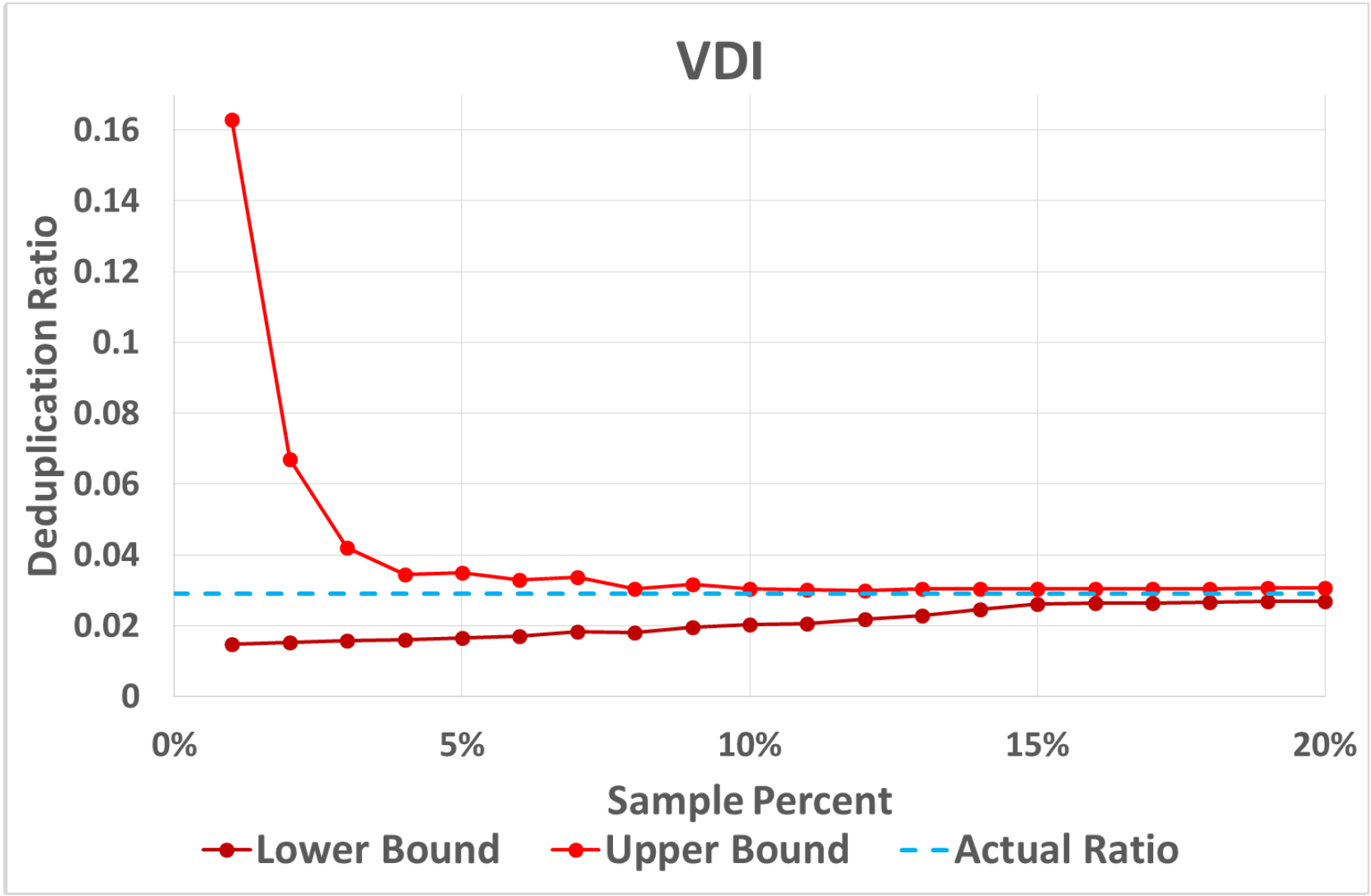


– includes low memory and reading at 1MB super-chunks

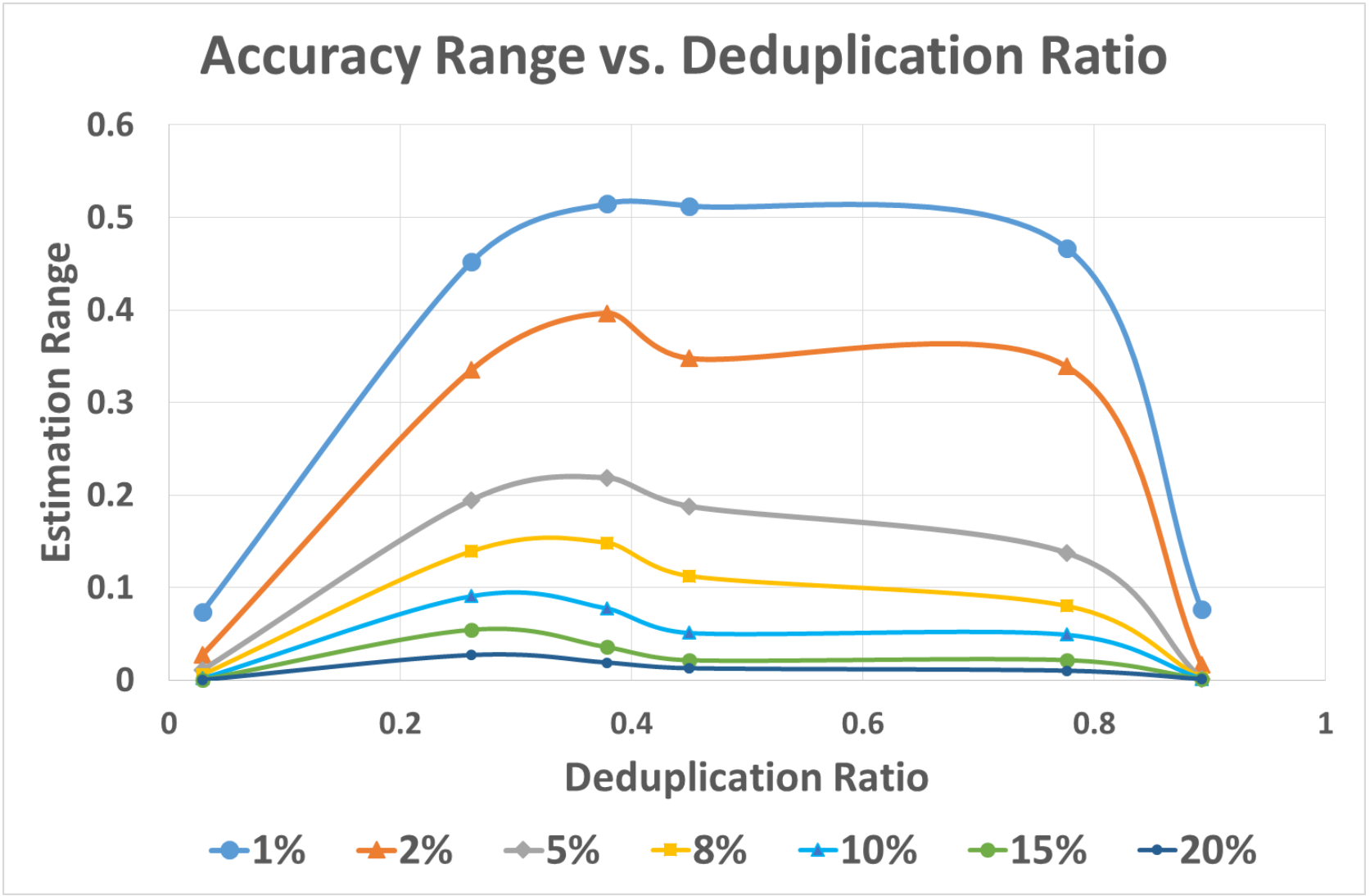


– includes low memory and reading at 1MB super-chunks

Putting it all together



– includes low memory and reading at 1MB super-chunks



- Can stop with 1-2% sample when dedupe ratio is **very high** or **very low**
- Otherwise need 8% or more

Summary

- **New fast, accurate, low memory estimation for data reduction**

- **Speed up of at least 3X over state of the art**
  - Assumes 15% sampling and data on rotating disks
  - Will be much faster (15X and more) when:
    - Reduction ratio is high or low – can stop early
    - Data on flash – faster random access

Thank You !

