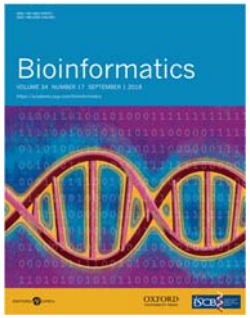
Binning directories: An efficient data structure to query k-mers in partitioned string sets



Temesgen H. Dadi, Enrico Siragusa, Vitor C. Piro, Andreas Andrusch, Enrico Seiler, Bernhard Y. Renard and Knut Reinert

Knut Reinert
Freie Universität Berlin

International Workshop on PDBs and Large Scale Search

Motivation

Freie Universität Berlin

- The indexing step in read mapping:
  - Computationally very expensive
  - But amortizes in time due to the static nature of reference genome e.g., hg38 since 2013
- Read mapping in metagenomics
  - A large set of reference genomes
  - Frequently changing (Not static)
  - Undermines the purpose of indexing

Motivation

Freie Universität Berlin

- Very large set of references

To build FM-Index

- ~100 GB RAM
- ~10-27 hours
- Updating takes the same amount of time because the index structures (CSA) are static

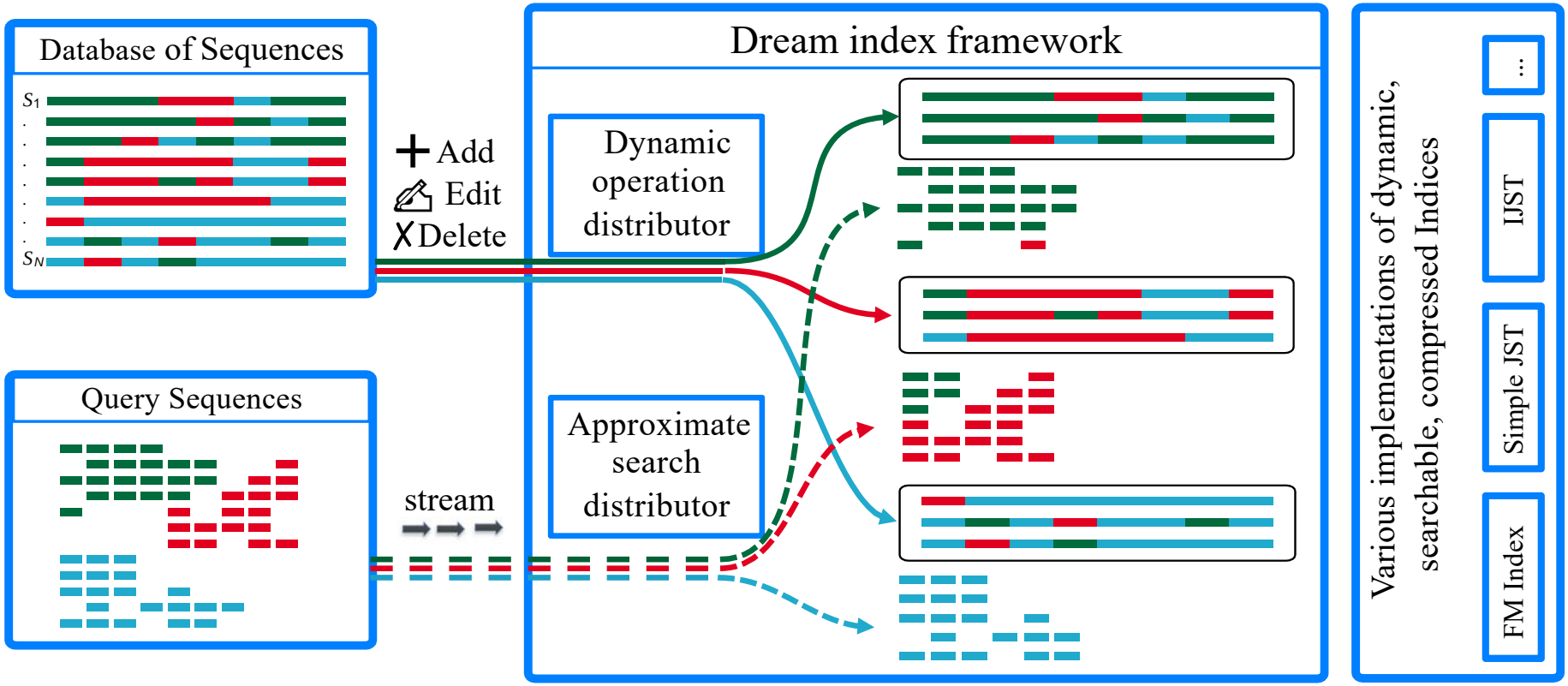
30 GB is "*not that big*"

➢ Partition the reference databases into smaller parts (bins)

DREAM Index framework

Freie Universität Berlin

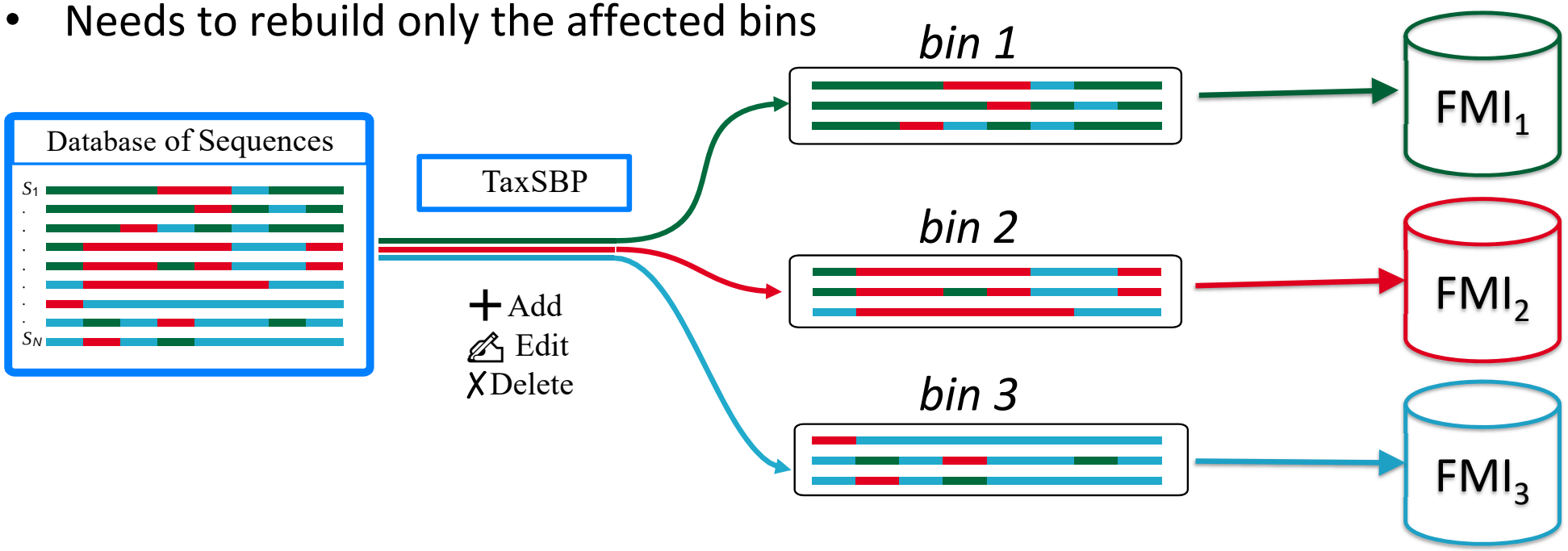
DREAM = Dynamic seaRchablE pArallel coMpressed index



Bin with TaxSBP and then index

Freie Universität Berlin

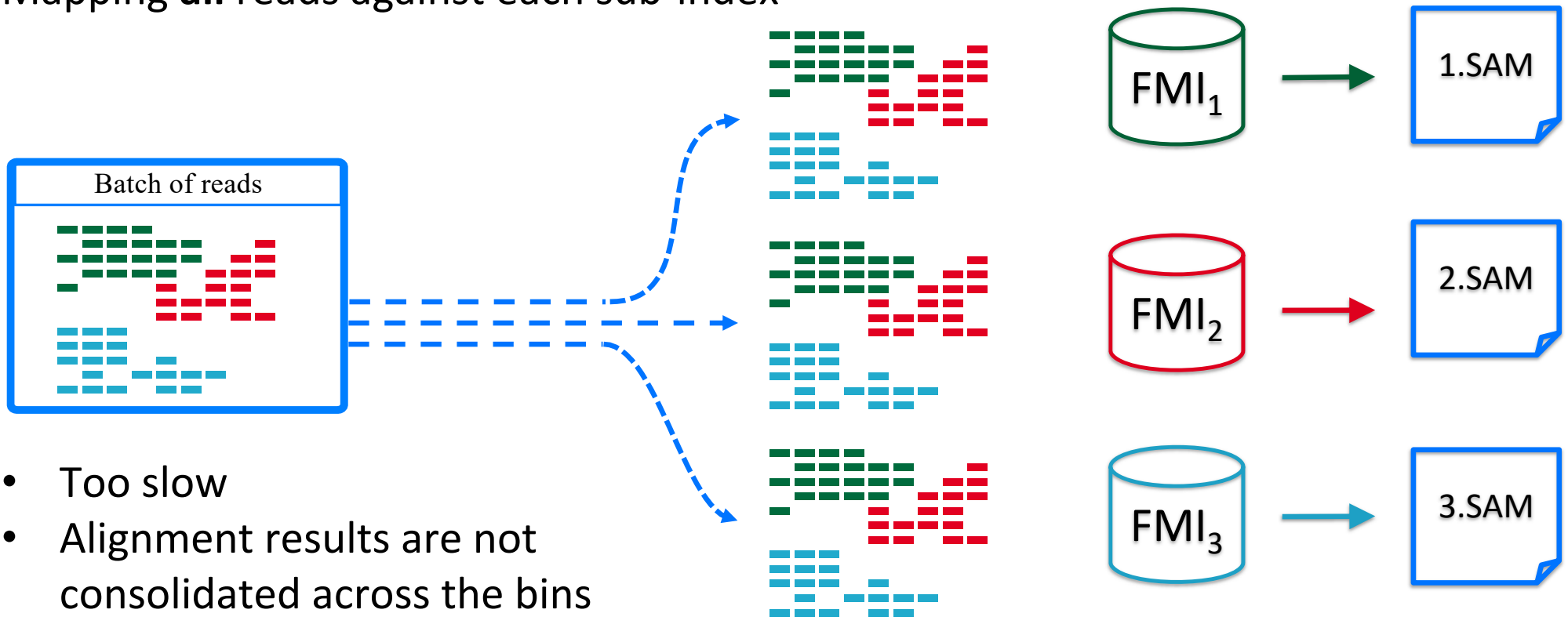
- Taxonomy based binning by V. Piro
- Hierarchically structured bin packing algorithm [*Codenotti et al.*]
- NCBI Taxonomy database
- Needs to rebuild only the affected bins



Trivial distributed mapping

Freie Universität Berlin

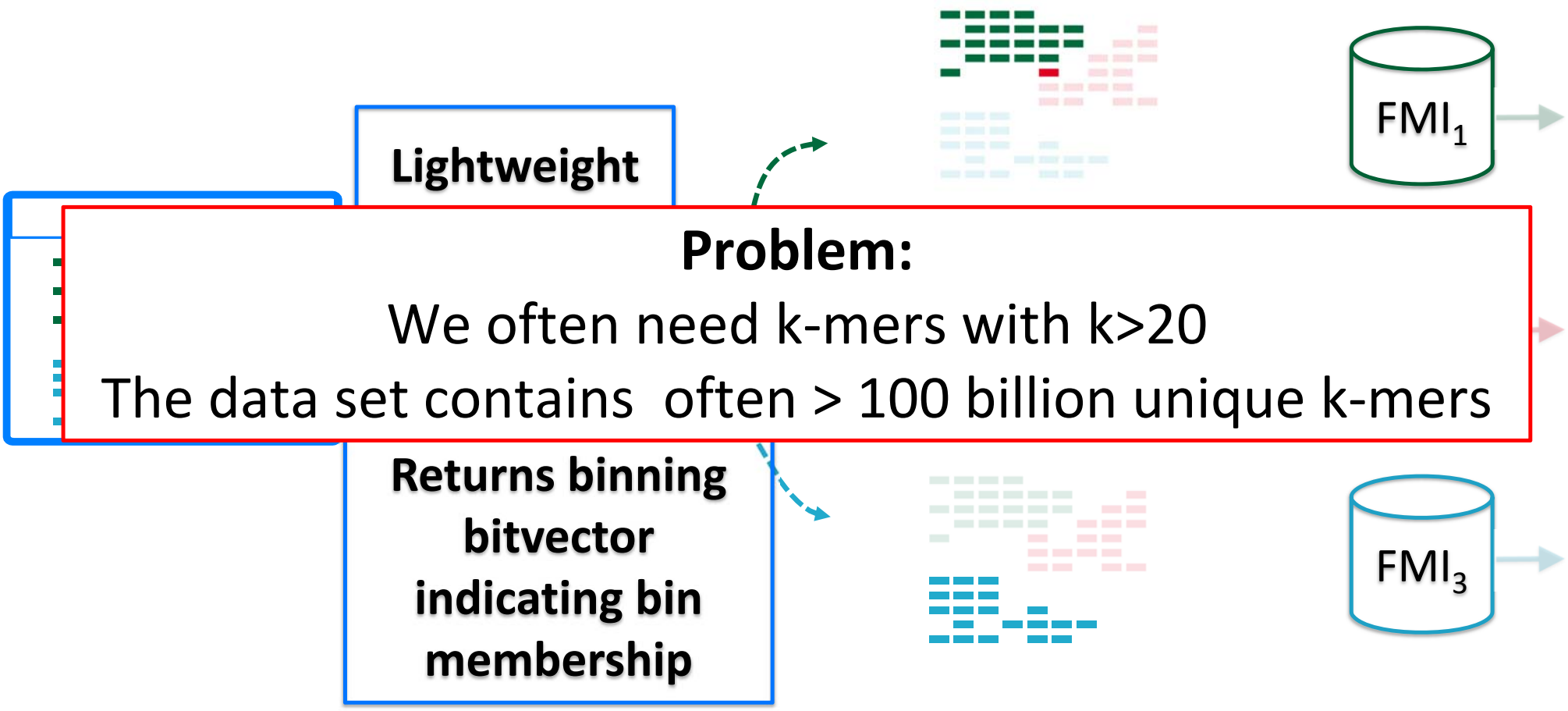
Mapping **all** reads against each sub-index



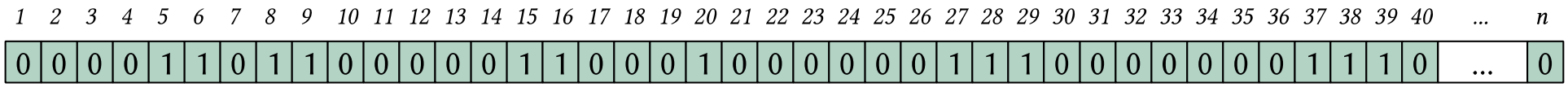
- Too slow
- Alignment results are not consolidated across the bins

Pre-filter reads before mapping

Freie Universität Berlin



Bloom Filter



- A bit-vector of size *n*
- *h* different hash functions - $\{H_1, H_2, \ldots H_h\}$
- $H_i : K \rightarrow L_i$ where $0 \leq L_i < n$
- To add a ***k-mer*** we simply set *h* bit positions
- During lookup, we expect all of the *h* bit positions to be set.
- Collisions are allowed in the expense of false positive answers.

$$P_{fp} = \left(1 - \left(1 - \frac{1}{n}\right)^{h*m}\right)^{h}$$

Where:
m = the number of elements ***Added***

Bloom Filter

Freie Universität Berlin

Why multiple hash functions?

Size of bit-vector (n) = 1 Billion

Number of hash functions

m (Billion)	1	2	3	4
1.0000000	0.632120549	0.747645060	0.857951631	0.928725749
0.5000000	0.393469332	0.399576389	0.468861712	0.558973136
0.2500000	0.221199212	0.154818116	0.146891590	0.159661290
0.1250000	0.117503098	0.048929091	0.030579353	0.023968649
0.0625000	0.060586936	0.013806977	0.004997657	0.002394056
0.0312500	0.030766765	0.003670777	0.000716668	0.000190633
0.0156250	0.015503563	0.000946594	0.000096030	0.000013475
0.0078125	0.007782062	0.000240360	0.000012431	0.000000896
0.0039063	0.003898630	0.000060560	0.000001581	0.000000058

Probability of false positive answer (FP rate)

Smaller FP rate &
Add more elements.

General principle:

$$\boldsymbol{h} \leq \frac{\boldsymbol{n}}{\boldsymbol{m}}$$

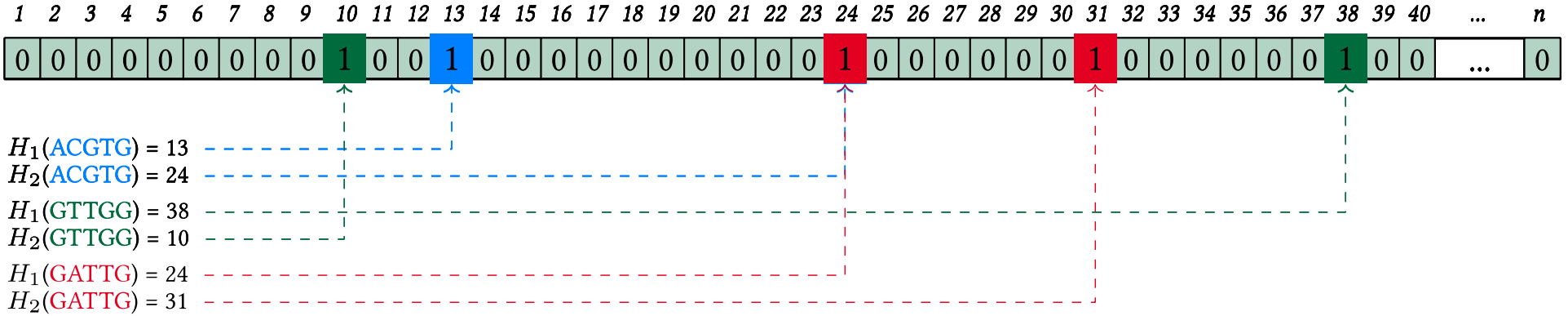
Bloom Filter

Freie Universität Berlin

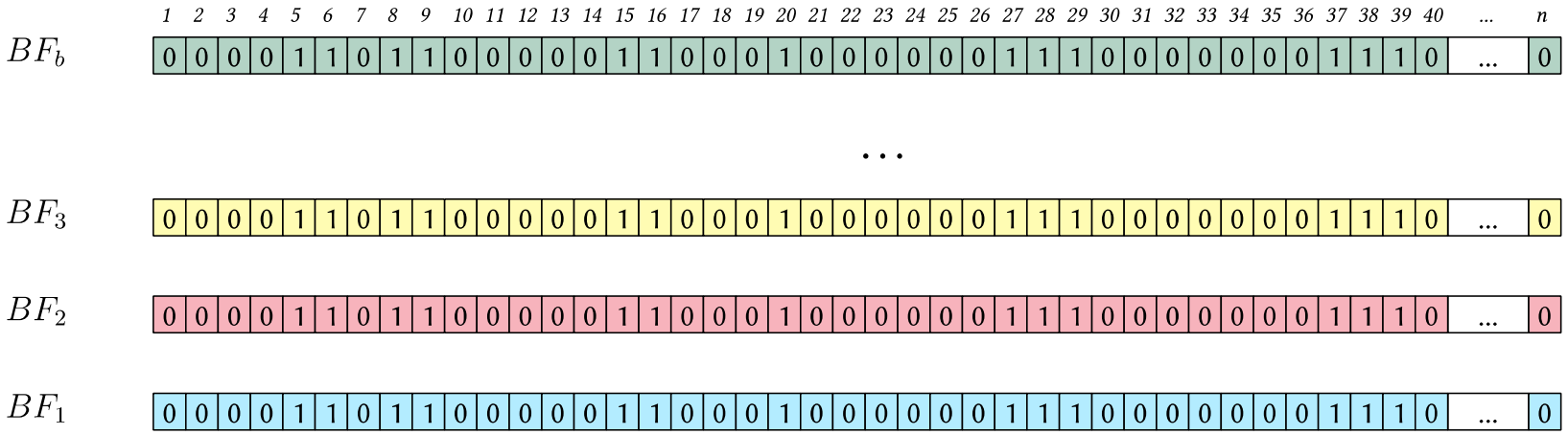
E.g. Add k-mers

ACGTG GTTGG GATTG

2 hash functions: {H_1, H_2}

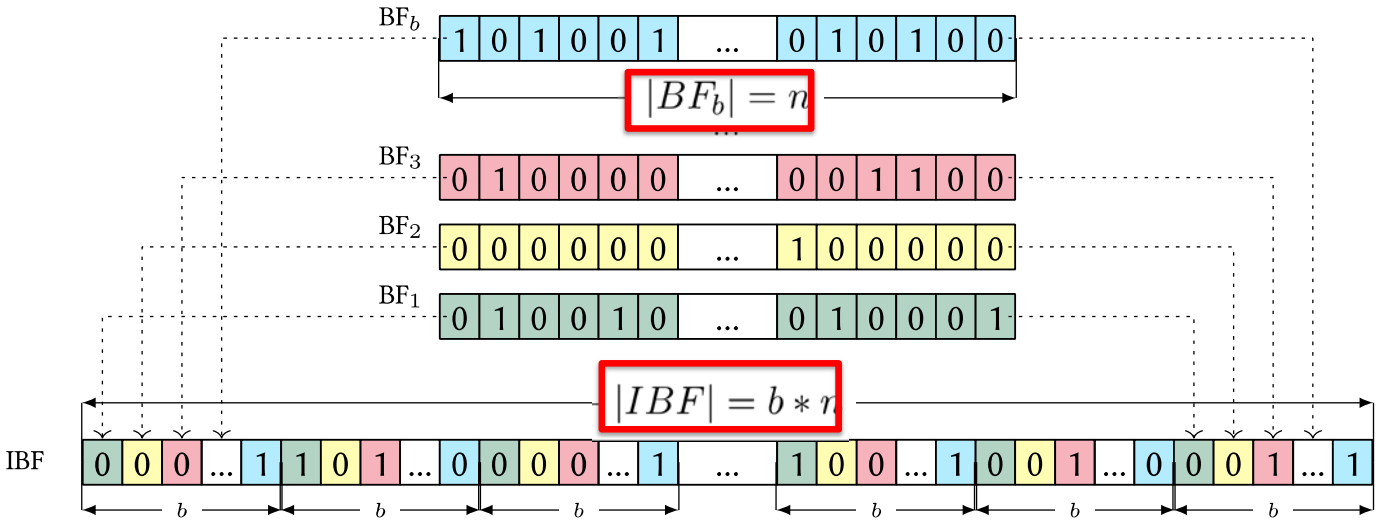


Bloom Filter for each bins?



- We need to check every k-mer of a read against each Bloom filter and construct binning vector

Interleaved Bloom Filter (IBF)



To add a k-mer (**K**) from bin **j**

We set position

$$\mathbf{b} \times H_i(\mathbf{K}) + \mathbf{j}$$

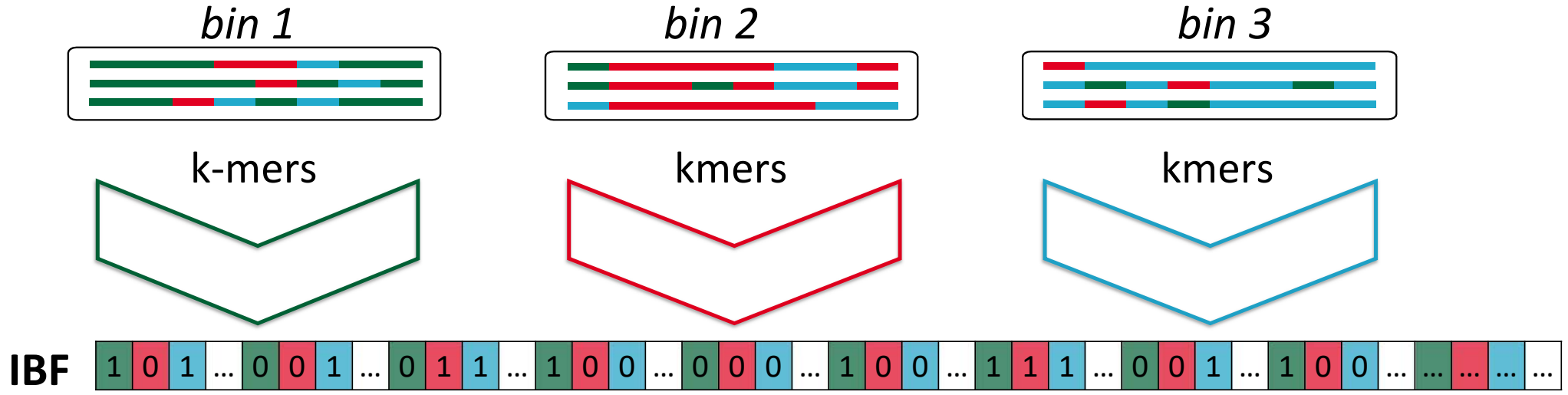
To check in which bins a k-mer is present we use the bit blocks

$$[H_i(\text{kmer}) \times \mathbf{b},\ H_i(\text{kmer}) \times (\mathbf{b}+1))$$

Interleaved Bloom Filter (IBF)

A light weight filtering data structure

- Based on bloom filters
- Contains the kmer information of all bins
- Together with q-gram lemma → very fast pre-filter
- 37 times faster than the trivial method (1024 bins)

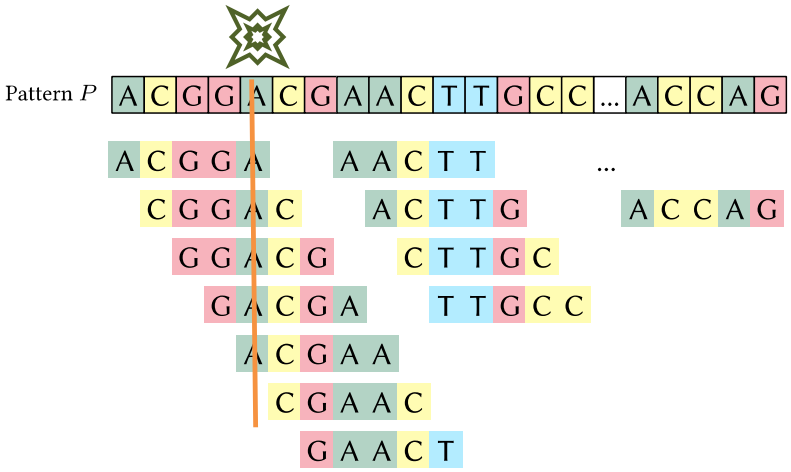


K-mer Counting Lemma

Freie Universität Berlin

For a given **k** and number of errors **e**, there are $k_p = |P| - k + 1$ many k-mers in pattern **P** and an approximate occurrence of **P** in **T** has to share at least $t = (k_p - k \cdot e)$ k-mers

Total number of k-mers



An error can destroy at most k k-mers
e errors can destroy **k·e** k-mers

Limitation when ***e*** is bigger
$t = |P| - (e+1) \cdot k + 1$

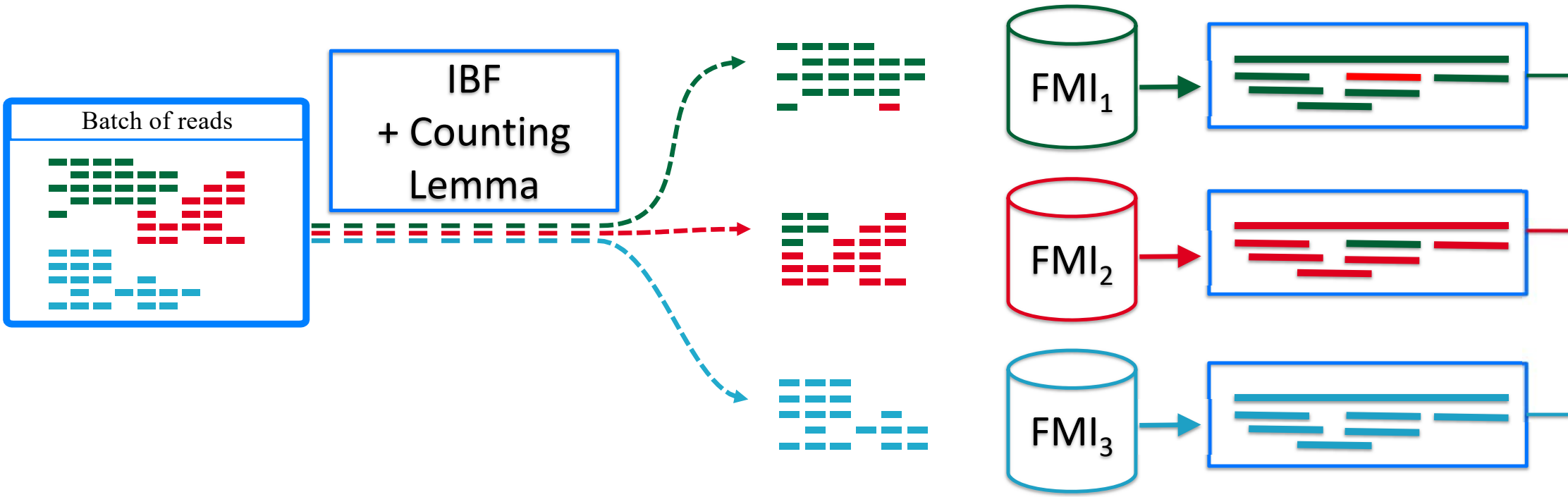
The Yara read mapper

Freie Universität Berlin

- State of the art read mapper of the SeqAn library
- Both **all mapping and strata mapping**
- Yara is a **fully sensitive** read mapper
- Fast (2X Bowtie2, 3X BWA-MEM)

DREAM-Yara *(Mapping)*

Freie Universität Berlin



Benchmark

Freie Universität Berlin

Infrastructure

- Intel(R) Xeon(R) CPU E5-2650 v3 2.30GHz processor
- 130GB of memory
- 8 threads whenever the tool allows

Dataset (2017-09-26)

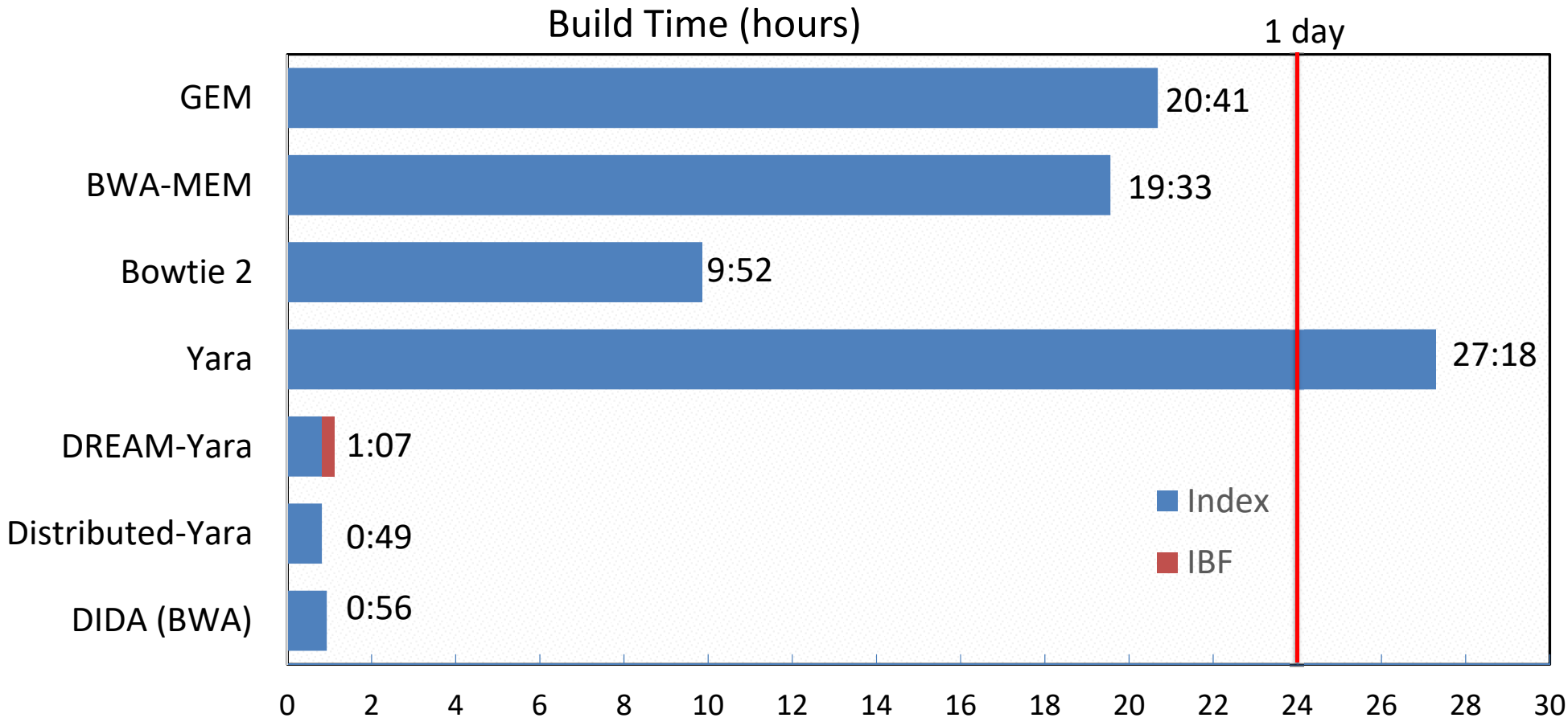
- 15,250 sequences
- 2,991 species
- 31.34 GB

Update (2017-12-19)

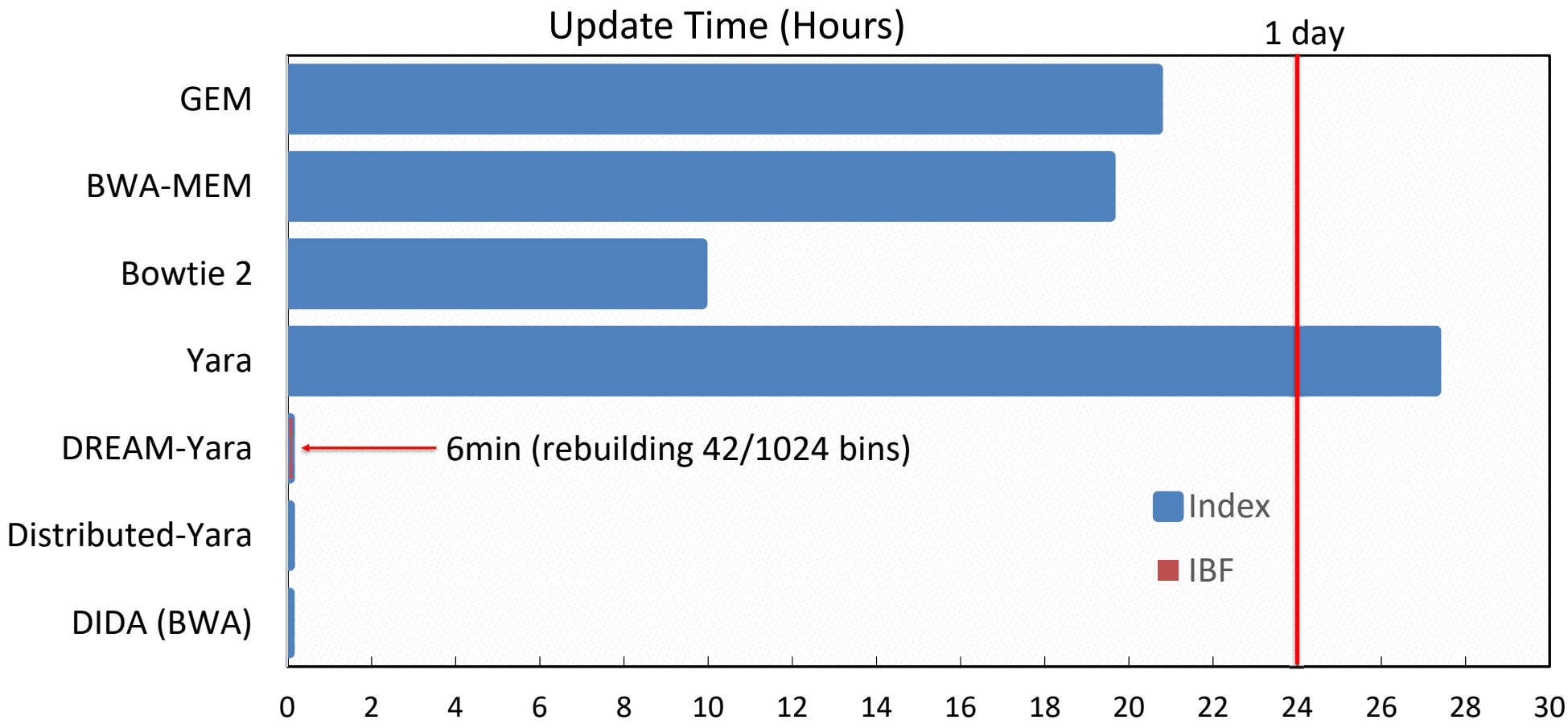
- 155 new sequences of E. coli
- 42/1024 bins affected
- 230 MB

BWA-MEM **Bowtie2** **GEM** **DIDA (BWA)**

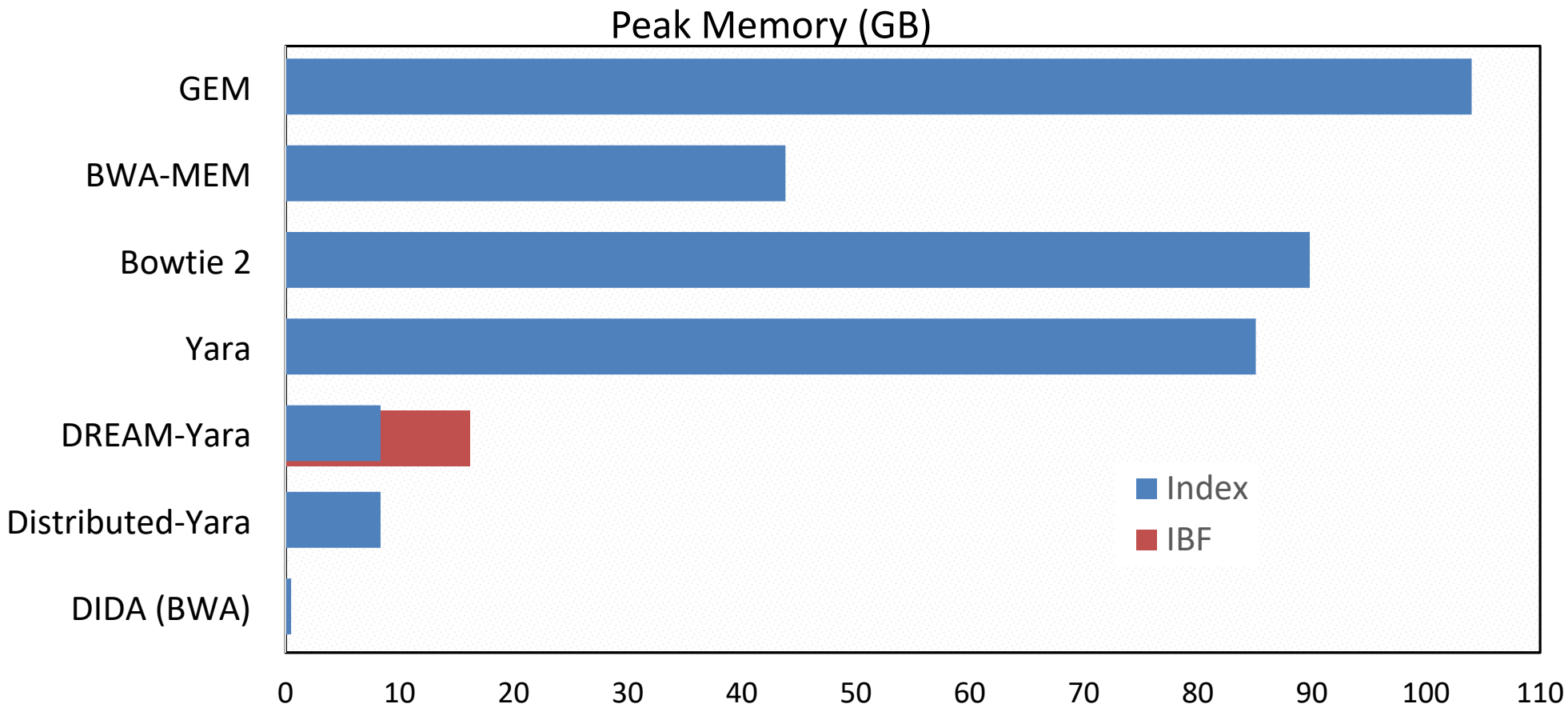
Indexing – build time



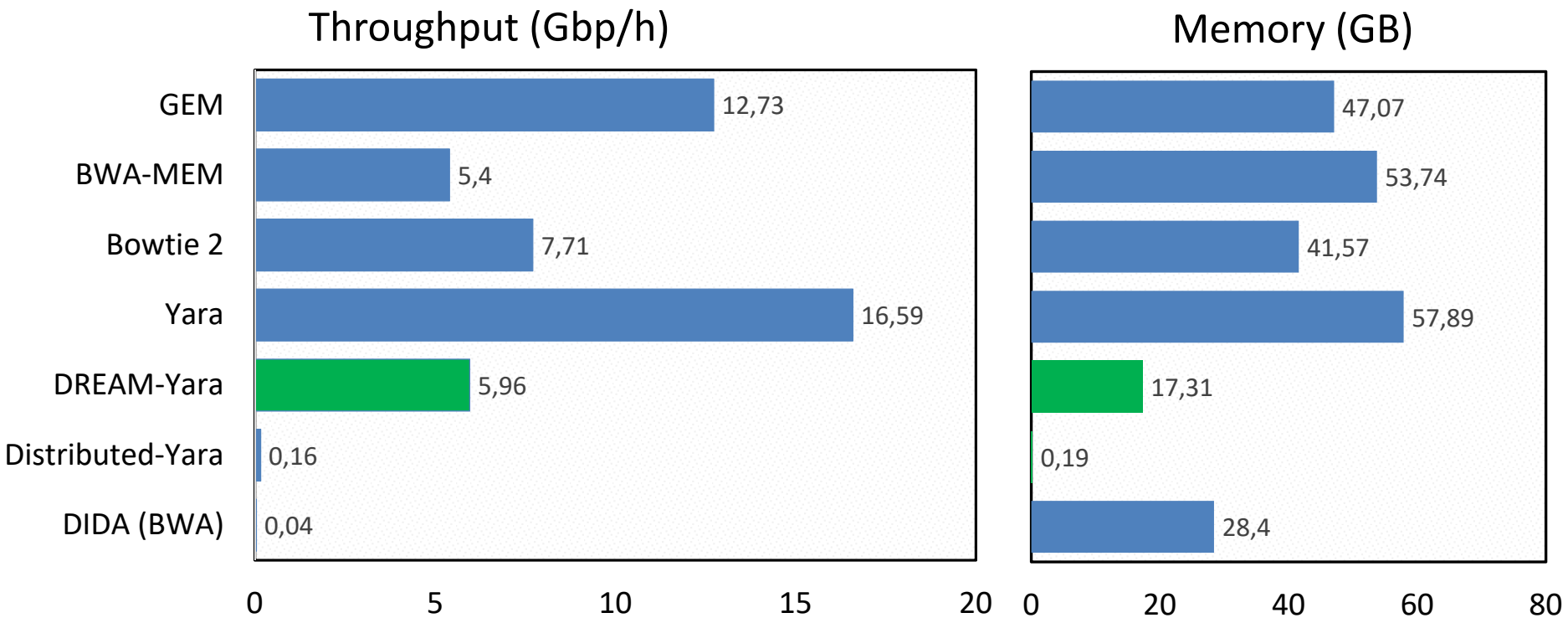
Indexing – update time



Indexing – memory footprint



Mapping speed and memory



Rabema benchmarks
(Sensitivity)

Freie Universität Berlin

Read Set: SRR6504858 (human gut metagenome sample)

	Co-opt. locations [% Normalized]				Co-opt. locations [% Absolu			
Distributed-Yara	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
		100.0	100.0	100.0		100.0	100.0	100.0
DREAM-Yara	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
		100.0	100.0	100.0		100.0	100.0	100.0
Yara	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
		100.0	100.0	100.0		100.0	100.0	100.0
GEM	99.8	100.0	100.0	99.9	96.4	100.0	99.8	95.4
		99.5	99.7	94.8		74.2	67.6	79.3
Bowtie 2	99.8	99.9	100.0	99.9	81.6	80.8	90.4	89.9
		99.7	99.5	97.3		67.6	63.4	81.3
BWA-MEM	99.3	100.0	100.0	99.8	85.2	83.6	92.1	93.6
		98.2	93.8	90.1		81.5	72.7	84.5
DIDA-BWA	99.5	100.0	100.0	99.8	85.2	83.6	92.1	93.6
		98.2	93.8	–		81.5	72.7	84.5

100.0 99.8 95.4
74.2 67.6 79.3

e less

Absolute:
All locations equally matter

Summary

Freie Universität Berlin

- DREAM framework for distributed read mapping which can support fast updates of the sub-indices.
- A fast and light weight pre-filter using IBF and q-gram lemma (q=19)
- DREAM-Yara, implementation of an in-memory distributed version of Yara.
    - Competitive mapping speed
      (faster than BWA and slightly slower the Bowtie2)
    - The pre-filter introduced a ~ 37X speedup.
    - It can conduct a typical batch index update in 6 minutes
      (Bowtie2 ≈ 10 hr; BWA & GEM ≈ 20 hr; Yara ≈ 27 hr)

Thank you!

Co-Authors

Enrico Siragusa,
Vitor C. Piro,
Andreas Andrusch,
Enrico Seiler,
Bernhard Y. Renard and
Knut Reinert

Freie Universität Berlin

Algorithmic Bioinformatics Group

SEQAN

@SeqAnLib

www.seqan.de

ROBERT KOCH INSTITUT

intel

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

zwanzig20
PARTNERSCHAFT FÜR INNOVATION

UNTERNEHMEN REGION
Die BMBF-Innovationsinitiative Neue Länder

International Max Planck Research School for Computational Biology and Scientific Computing

Outline

Freie Universität Berlin

- **Introduction**
  - Motivation
  - The DREAM index framework
- **Methods**
  - Binning sequences
  - Interleaved Bloom Filter
  - The Yara read mapper
  - DREAM-Yara
- **Evaluation**
  - Index creation and updating benchmark
  - Rabema benchmarks (Sensitivity)
  - Read mapping speed

The Yara read mapper

Freie Universität Berlin

- State of the art read mapper of the SeqAn Library
- Yara is an **exact** read mapper (**All mapper**)
- Fast (2X Bowtie2, 3X BWA-MEM)
- Implements strata based mapping

E.g. Mapping a read (*100 bp long*) up to 5 errors (*edit distance*)

Best map has 2 errors

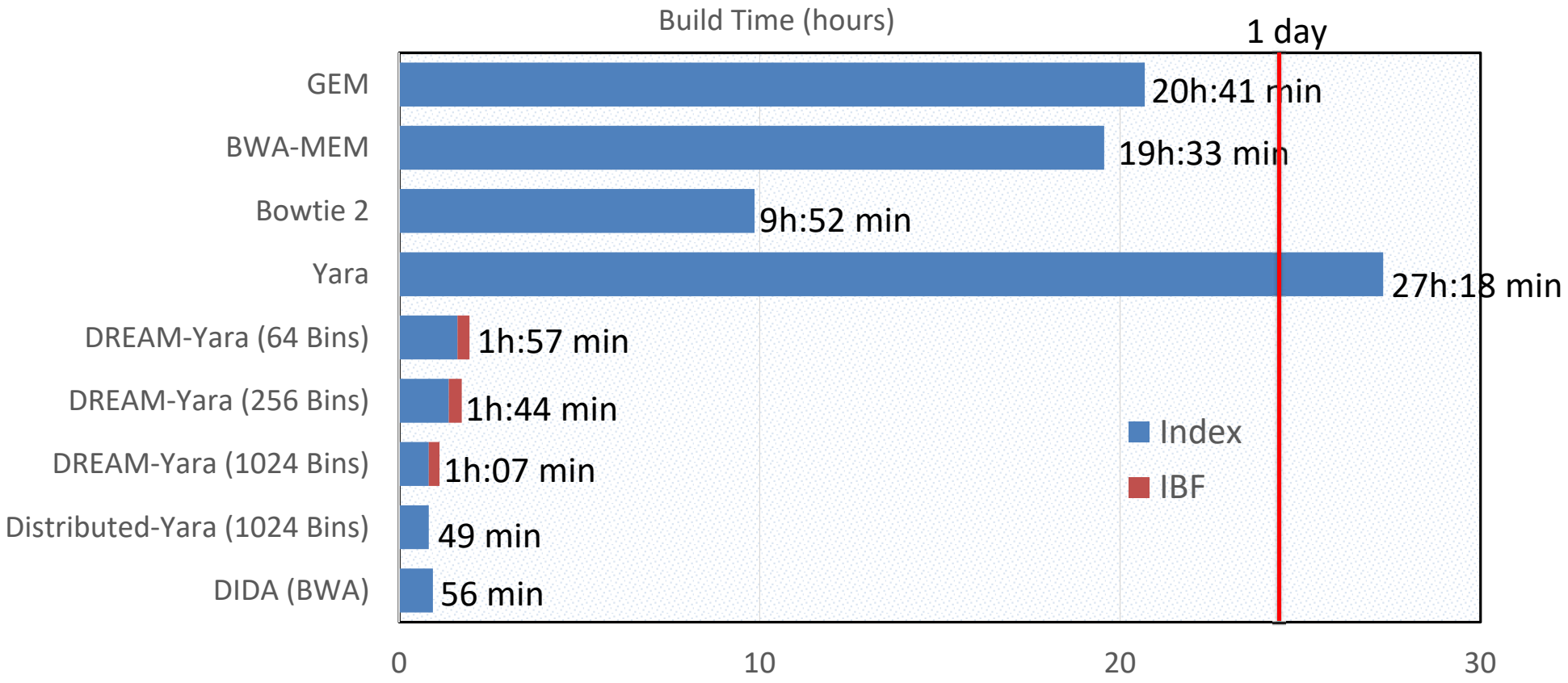
0 strata => 2e

1 strata => 2e, 3e

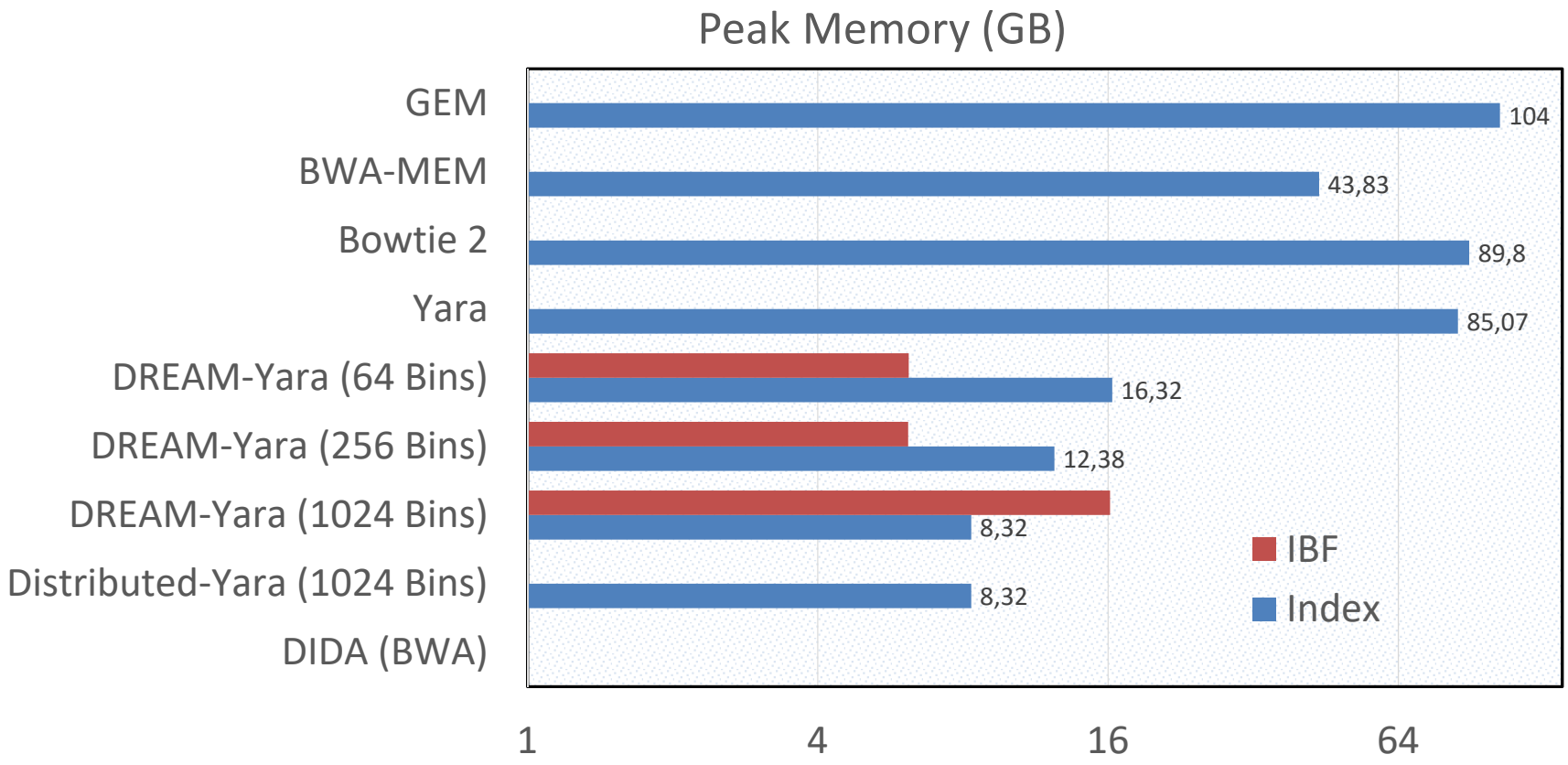
2 strata => 2e, 3e, 4e

3 strata => 2e, 3e, 4e, 5e

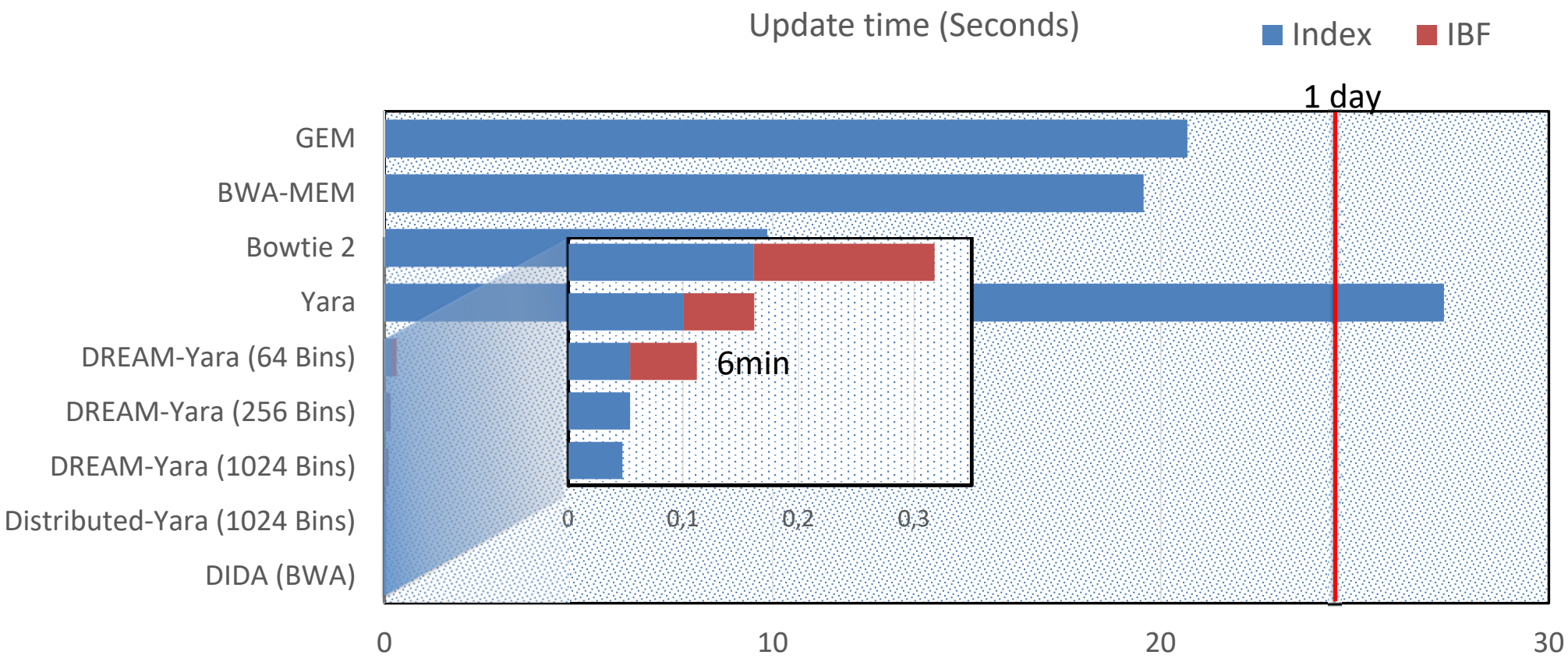
Benchmark (Index creation)



Benchmark (Indexing - memory)



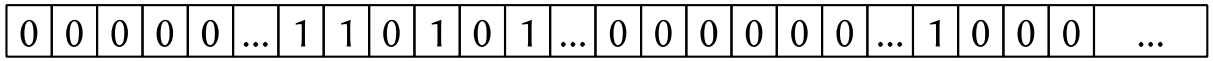
Benchmark (Index updating)



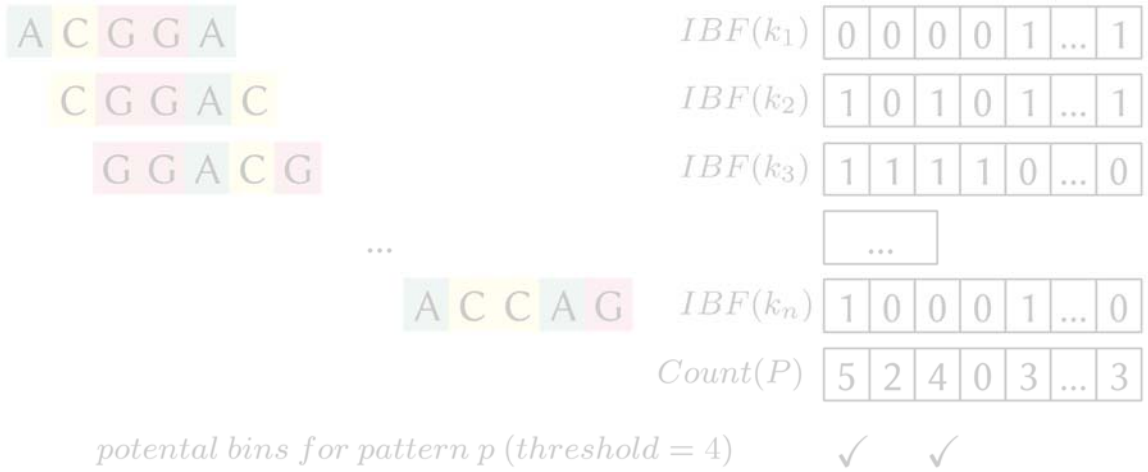
IBF + Counting Lemma

Freie Universität Berlin

Interleaved Bloom Filter IBF

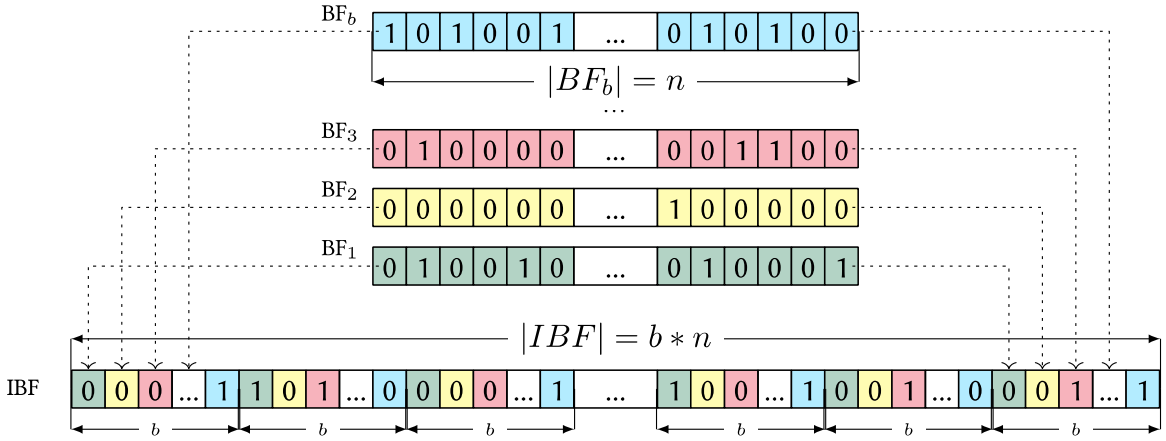


Pattern p



Approximate search distributor

IBF Caveats



Works best when

- Bins are equal in text size