The International Workshop On
DISCS
Data-Intensive Scalable Computing Systems

Par-BF: a Parallel Partitioned Bloom Filter for Dynamic Data Sets

Yi Liu[1], **Xiongzi Ge**[2], David H.C. Du[2], and Xiaoxia Huang[1]

[1]Shenzhen Institutes of Advanced Technology, CAS

[2]Computer Science and Engineering, University of Minnesota

Center for Research in Intelligent Storage

Outline

- Brief Summary of Bloom Filters (BFs)
- BF Design for Dynamic Data Sets
- Par-BF
- Evaluation
- Summary

Bloom Filter (BF)

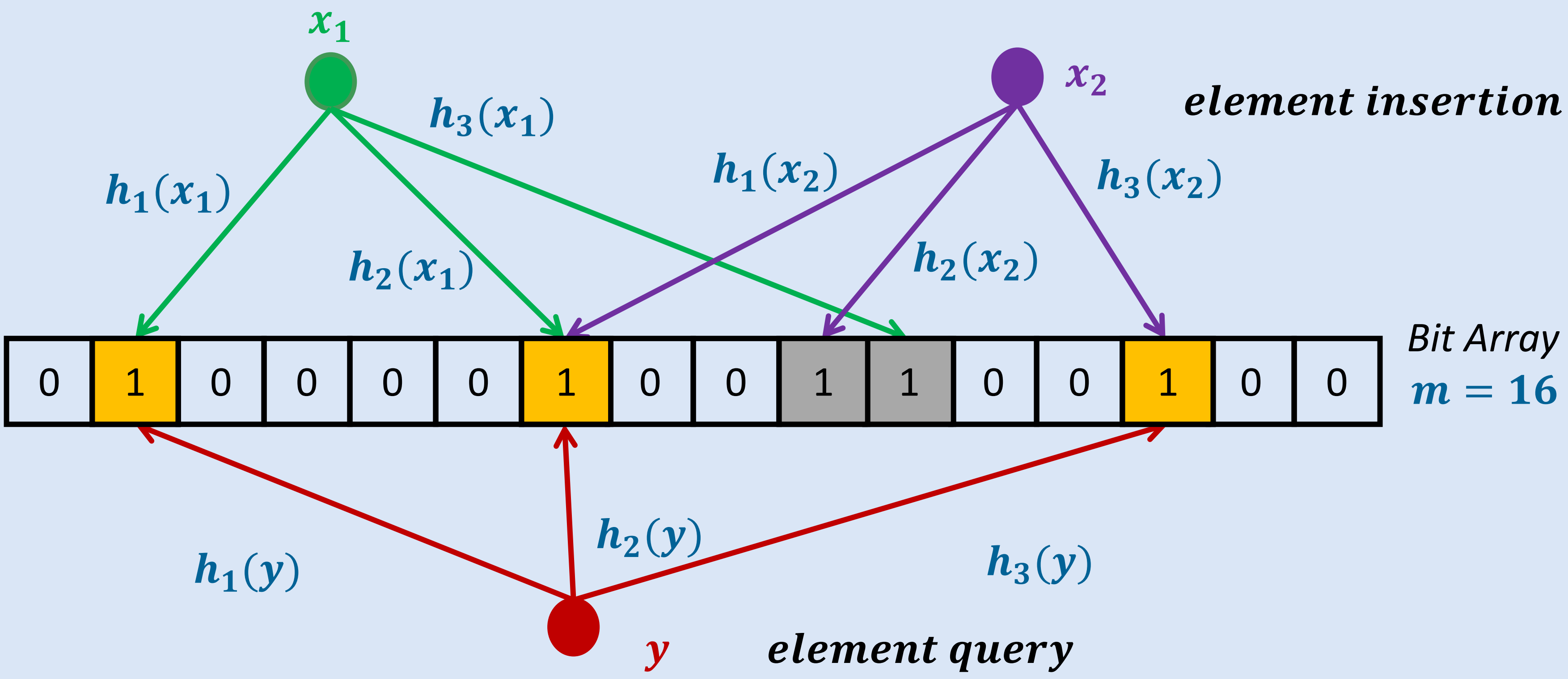
- A space-efficient index to quickly answer "Is an element x in the target set S ?"
- Widely used as an in-memory index

> *"Wherever a list or set is used, and space is at a premium, consider using a Bloom filter if the effect of false positives can be mitigated."*
>
> --A. Broder and M. Mitzenmacher
>
> *Network applications of bloom filters: A survey*

BF Definition

- A BF is an array of $\boldsymbol{m}$ bits representing a set $S = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_n}\}$ of $\boldsymbol{n}$ elements
  - *All bits are set to 0 initially*
- k independent hash functions $\boldsymbol{h_1}, \dots, \boldsymbol{h_k}$,with range $\{\boldsymbol{1}, \boldsymbol{2}, \dots, \boldsymbol{m}\}$
  - *Assume that each hash function maps each item in the universe to a random number uniformly over the range*
- For each element x in S, the bit position $\boldsymbol{h_i}(\boldsymbol{x})$ is set to 1, for $1 \leq i \leq k$
  - *A bit in the array may be set to 1 multiple times for different elements*

An example of BF



0	1	0	0	0	0	1	0	0	1	1	0	0	1	0	0

False Positive has been unexpected encountered that y is not existed in S but reports it is in!

Dynamic Data Sets

- Multiple disjointed and independent sets competing for a limited and shared pre-allocated space
- Cannot predict the number of elements in each set in advance

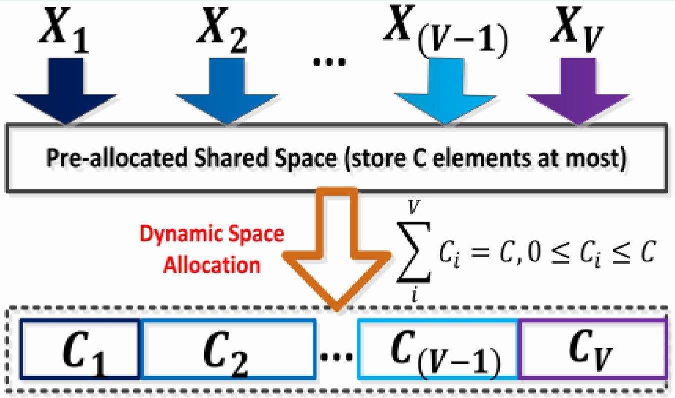

X_1
X_2
…
$X_{(V-1)}$
X_V
Pre-allocated Shared Space (store C elements at most)
Dynamic Space Allocation
$\sum_{i}^{V} C_i = C, 0 \leq C_i \leq C$
C_1
C_2
…
$C_{(V-1)}$
C_V

Figure 1: The size allocated to each X_i may change dynamically when V disjointed and independent sets $\{X_1, X_2, ..., X_V\}$ to compete for a limited and shared pre-allocated space.

Designing a BF for Dynamic Data Sets

- Dividing the shared BF into a certain number of fixed-size sub-BF units (much finer granularity)
- A new sub-BF is added into the sub-BF list of a set X_i for new insertions when all the previous sub-BFs are full
- Similar with the memory paging policy that each page size is usually 4KB

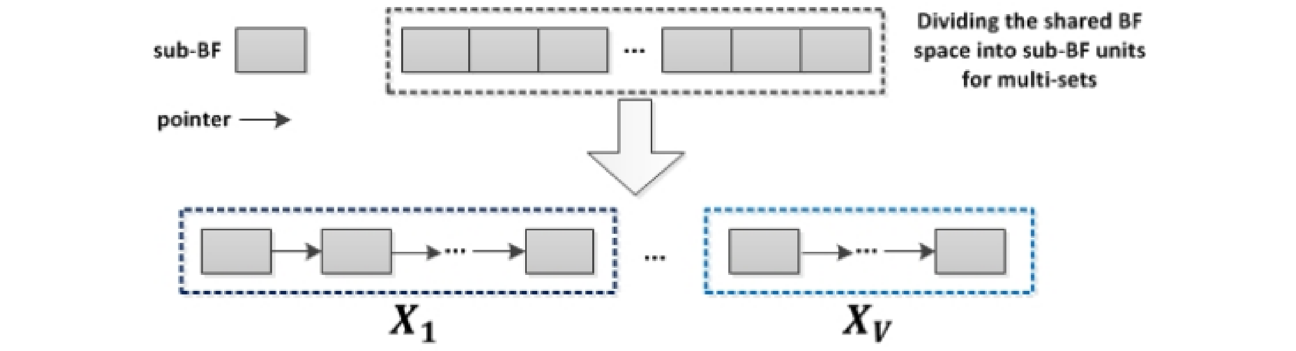
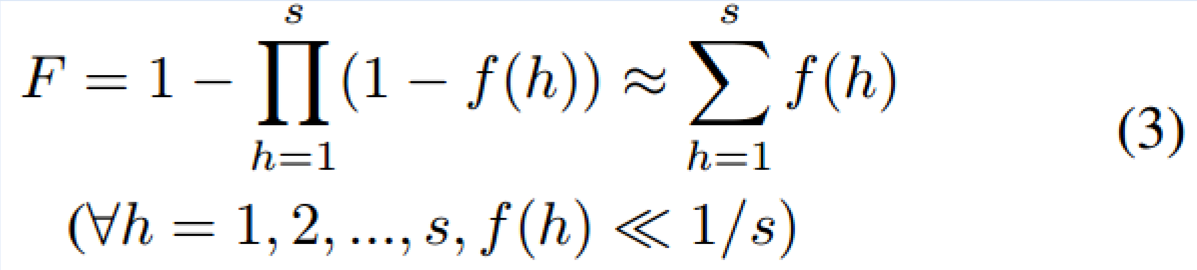

sub-BF
pointer
Dividing the shared BF space into sub-BF units for multi-sets
...
X_1
X_V

Figure 2: The sub-BF unit is the fundamental BF-based data structure for supporting dynamic sets.

Previous work (1)
The Dynamic Bloom Filters (DBF)

- A DBF consists of s homogeneous standard (or counting) sub-BFs
  - “Homogeneous” means both the array size $\boldsymbol{m}$ and the k hash functions are exactly the same
- Merit: support useful bit vector based algebra operations: *union, intersection*, and *halving*
- Defect: no mechanism to control the overall false positive rate, F



$$F = 1 - \prod_{h=1}^{s}(1 - f(h)) \approx \sum_{h=1}^{s} f(h) \qquad (3)$$

$$(\forall h = 1, 2, ..., s, f(h) \ll 1/s)$$

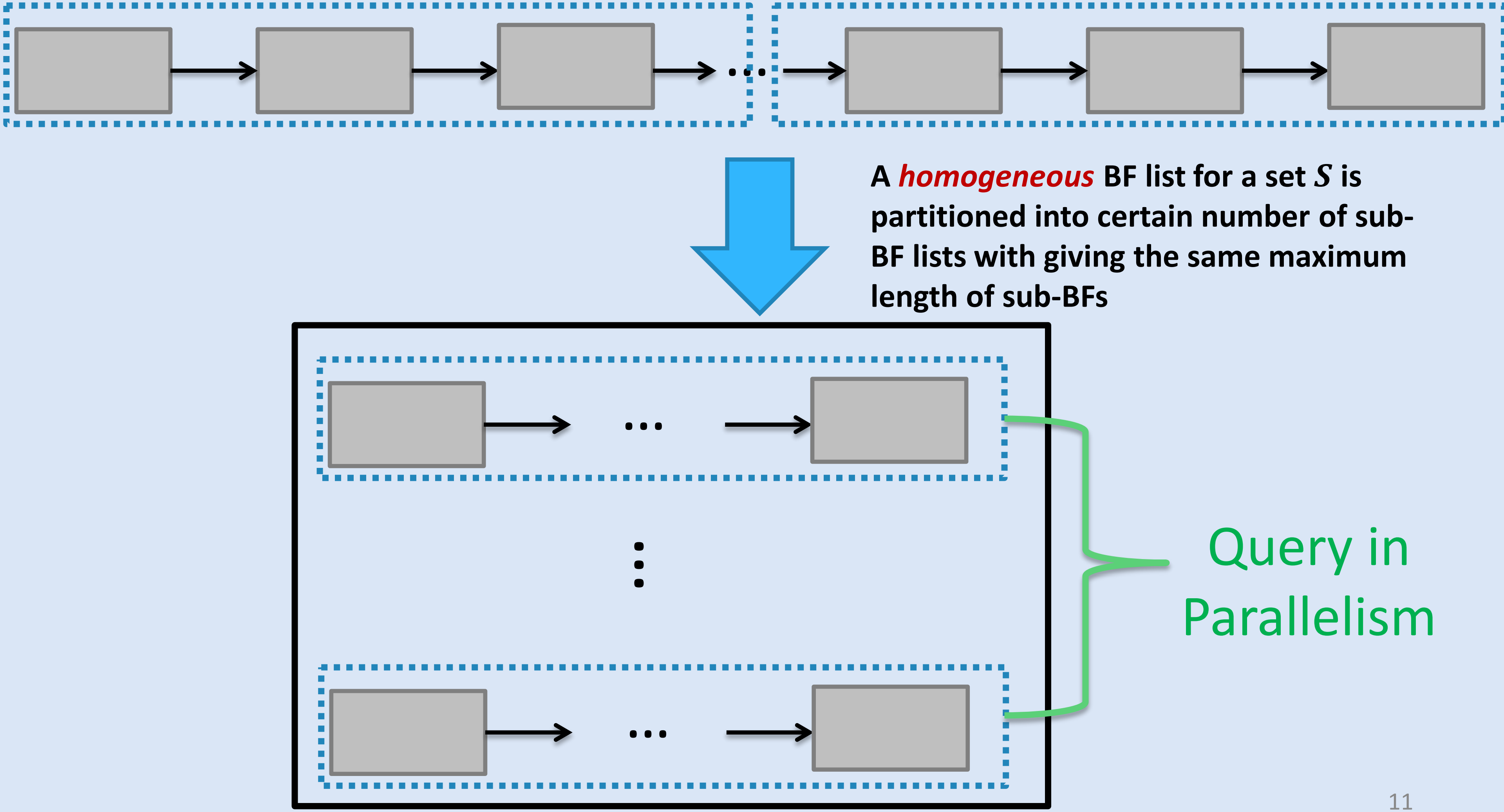
Previous work (2)
The Scalable BF (SBF)

- A SBF is made up of a series of heterogeneous sub-BFs
  - ➢ "heterogeneous" means both the size $\boldsymbol{m}$ and the k hash mapping functions of each sub-BF are different
- The key idea is that both $\boldsymbol{m}$ and k of each sub-BF is well-conducted with a tighter maximum fpp on a geometric progression
- Merit: the overall false positive rate, F is in convergence
- Main Defect: No support of BF-based algebra operations for simplifying resource management

Contributions of Par-BF

- Query in **Parallelism** on a sub-BF list basis
- Making the overall false positive rate F limited
- Supporting useful bit vector based algebra operations
- Performance-driven Initialization

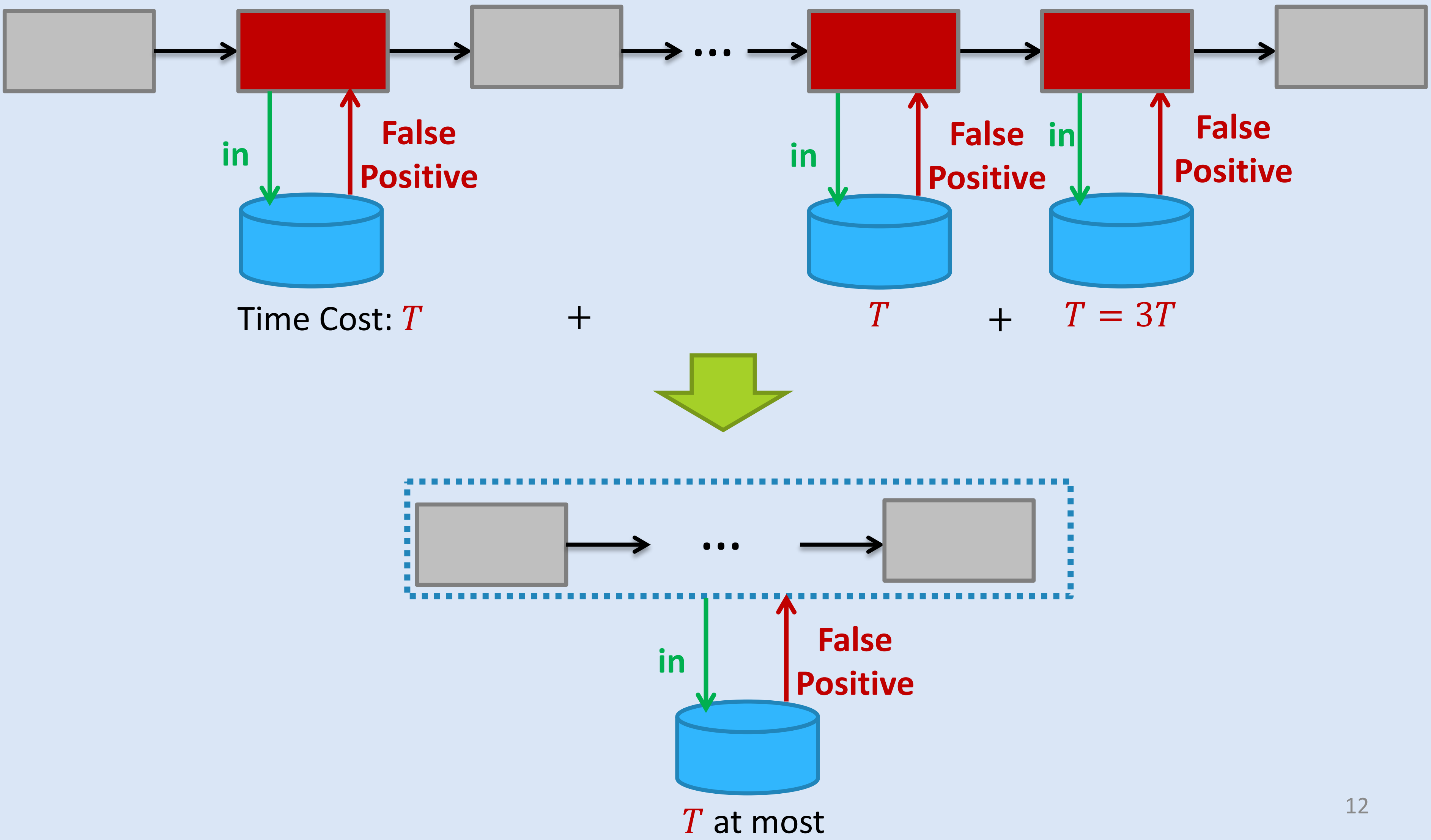
Partition Method



A ***homogeneous*** **BF list for a set S is partitioned into certain number of sub-BF lists with giving the same maximum length of sub-BFs**

Query in Parallelism

Merits of the Partition



Merits of the Partition

- Query in Parallelism makes fpp of an independent sub-BF list, F, in a limited range
- Supporting useful bit vector based algebra operations, since all sub-BFs are homogeneous

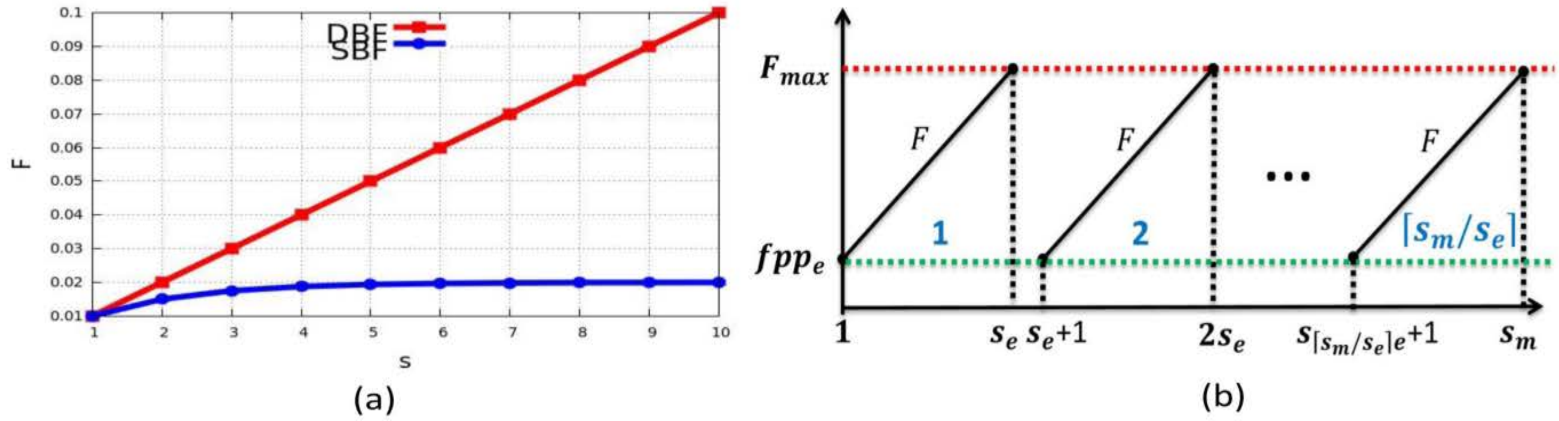


Figure 3: (a) An example of F value growth comparison between DBF and SBF, $f(1) = 0.01$, $r = 0.5$ in SBF. (b) Our recommended Par-BF design, $s_m = \frac{C}{n}$.

Performance-driven Initialization

- The thread is assigned on a sub-BF list basis to do independent and parallel membership query
  - Assuming the thread number can be maximally equal to that of CPU cores
- Guaranteeing the worst lookup cost of each thread,Q_{worst}, when the element x is not in the set S
- Guaranteeing the memory overhead is not exceed the expected size, M

The Worst Lookup Cost of each Thread

- If we define the expected look up cost, Q_{worst}, in advance, how to measure the upper bound value of fpp of each sub-BF list, F_{max}?
- $S_q \gg S_b$, when each sub-BF of the sub-BF list is in memory

$$\begin{aligned} Q_{worst} &= s_e(0.5^k \cdot (S_b + S_q) + (1 - 0.5^k) \cdot S_b) \\ &= \underbrace{F_{max} \cdot S_q}_{\text{Average lookup cost of an element in backup container}} + \underbrace{s_e \cdot S_b}_{\text{Average lookup cost in a sub-BF}} \end{aligned} \tag{7}$$

Initialization Results

- If we previously define the whole capacity space C, the thread number T, the I/O metrics: S_q, S_b and Q_{worst}, the maximum memory overhead M, all essential parameters of par-BF can be properly initialized.

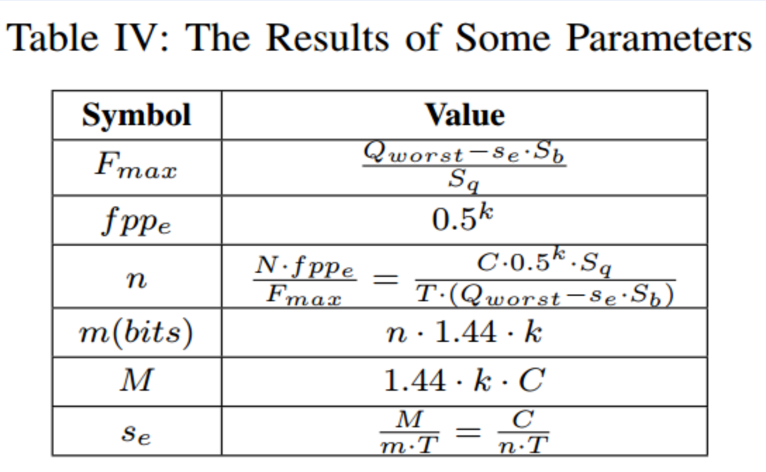


Table IV: The Results of Some Parameters

Symbol	Value
F_{max}	$\frac{Q_{worst} - s_e \cdot S_b}{S_q}$
fpp_e	0.5^k
n	$\frac{N \cdot fpp_e}{F_{max}} = \frac{C \cdot 0.5^k \cdot S_q}{T \cdot (Q_{worst} - s_e \cdot S_b)}$
$m(bits)$	$n \cdot 1.44 \cdot k$
M	$1.44 \cdot k \cdot C$
s_e	$\frac{M}{m \cdot T} = \frac{C}{n \cdot T}$

Known in advance

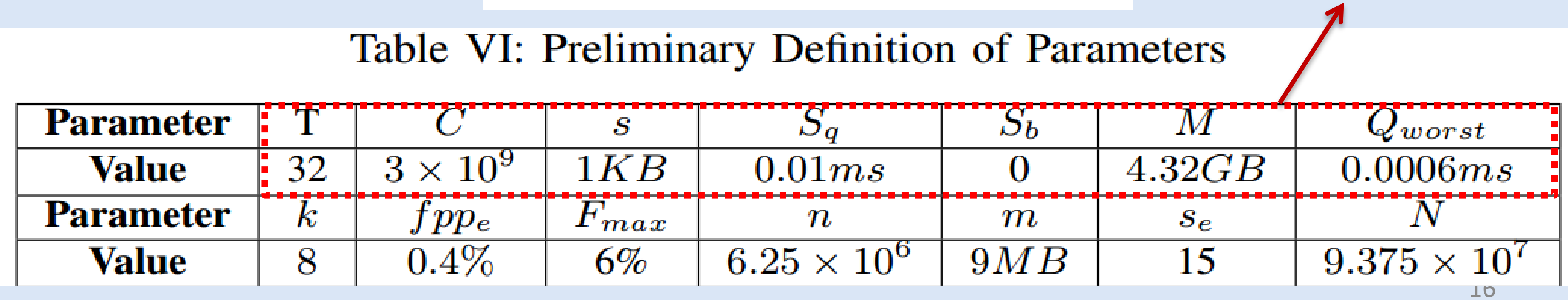


Table VI: Preliminary Definition of Parameters

Parameter	T	C	s	S_q	S_b	M	Q_{worst}
Value	32	3×10^9	$1KB$	$0.01ms$	0	$4.32GB$	$0.0006ms$
Parameter	k	fpp_e	F_{max}	n	m	s_e	N
Value	8	0.4%	6%	6.25×10^6	$9MB$	15	9.375×10^7

Evaluation

- Par-BF can find the sweet point, to balance the trade-off between high-performance and low-overhead
- The actual Q_{worst} is close to the expected Q_{worst} (nearly 99%) which validate our policy of parameter tunings

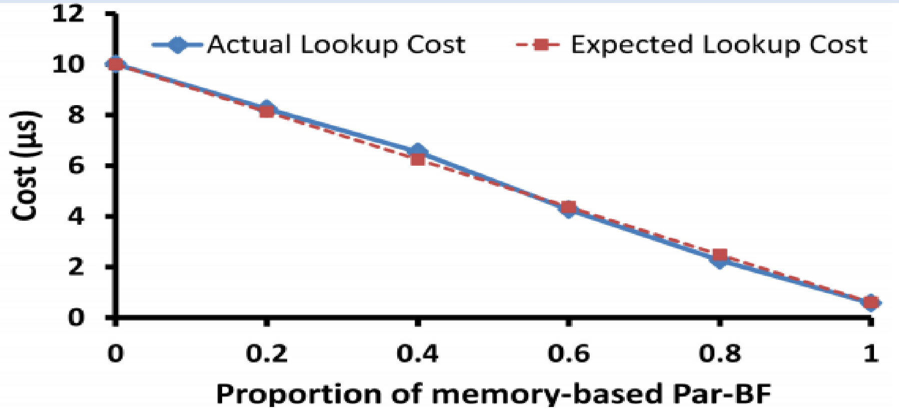


Figure 5: Q_{worst} comparisons between the actual value and the expected value when choosing different proportion of memory overhead ρ. *Q_{worst} is taken place when the trace data from **T2** are only firstly written to the K-V store. The expected Q_{worst} is calculated by $\rho \cdot S_b + (1 - \rho) \cdot S_q$.*

The Performance of Par-BF

- We record the average read throughput during running the three data traces T1, T2, and T3
- The IOPS of Par-BF outperforms that of DBF and SBF, from 6X to 10X and from 2X to 4X, respectively

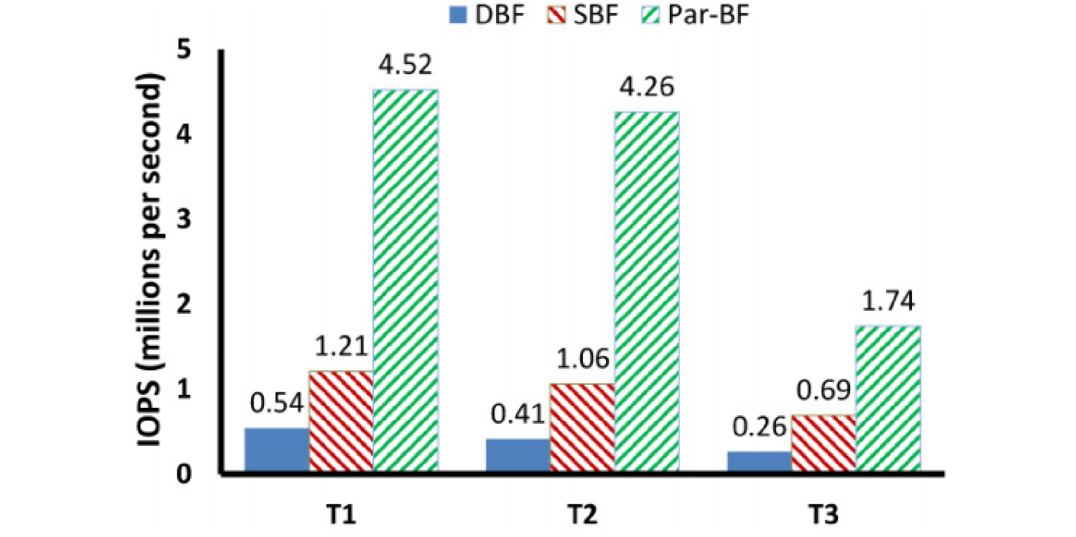


Figure 6: The throughput comparisons between DBF, SBF, and Par-BF through running T1, T2, and T3. *The initialized* $r = 0.5, s = 2$, *and* $f(1) = fpp_e$ *of SBF.*

Garbage Collection

- The memory overhead of both DBF and Par-BF is less than that of SBF by about 0.18GB
- the GC process which is only supported by *union operation between homogenous sub-BFs* makes the memory space more efficient, such as reducing the memory overhead by 0.45X in T2

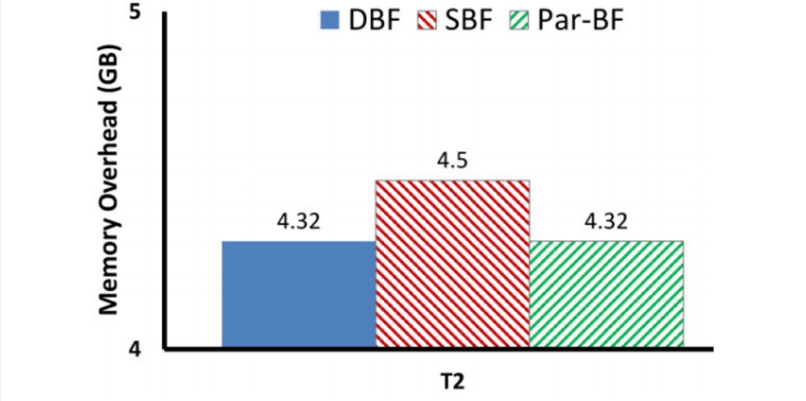


Figure 7: The memory overhead of using data trace **T2** by the three BF policies.

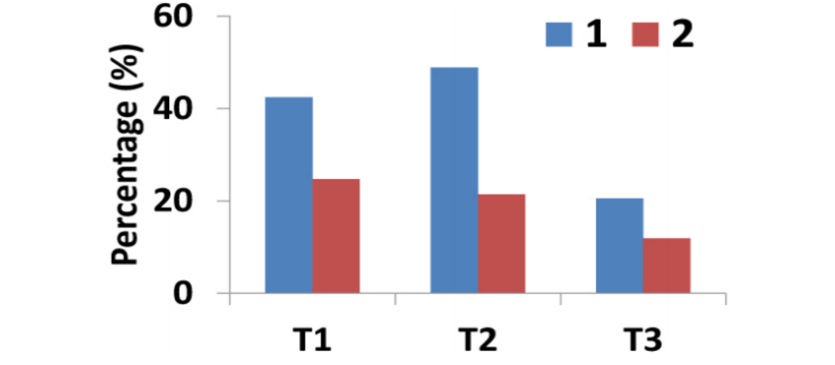


Figure 8: The comparison on percentage of chunk access in each trace (Type "**1**" denotes "**do GC**" and type "**2**" denotes "**do NOT GC**").

Summary and Future work

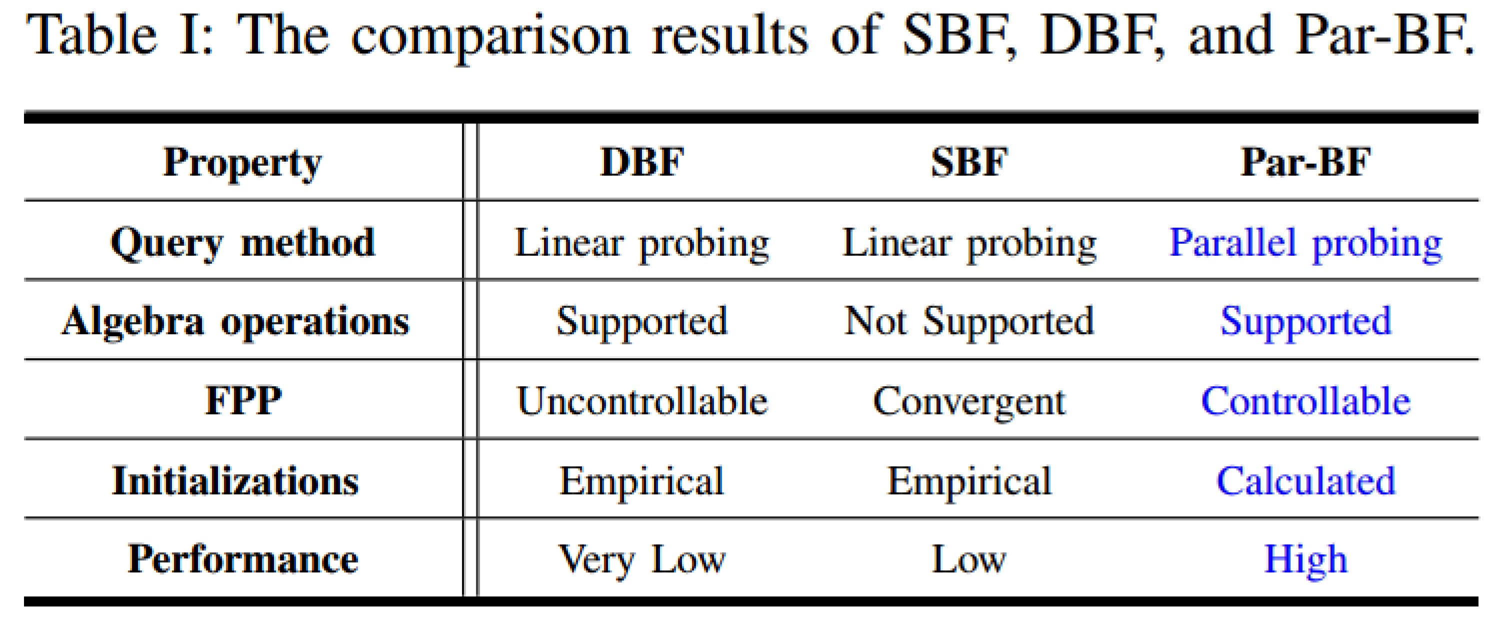


Table I: The comparison results of SBF, DBF, and Par-BF.

Property	DBF	SBF	Par-BF
Query method	Linear probing	Linear probing	Parallel probing
Algebra operations	Supported	Not Supported	Supported
FPP	Uncontrollable	Convergent	Controllable
Initializations	Empirical	Empirical	Calculated
Performance	Very Low	Low	High

Future Work:

1 Model improvement

2 Experiments with scientific computing applications

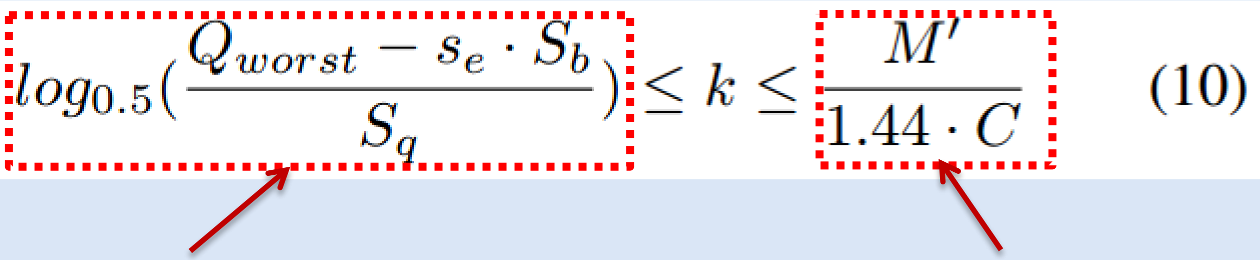
3 Experiments with different thread number when using different CPU cores

Thanks

Questions?

Limiting the k value in range

- If the maximum space overhead is limited by M', the inequality $k \le \frac{M'}{1.44 \cdot C}$ must be satisfied
- F_{max} must be greater than the expected fpp of a sub-BF, fpp_e,thus, the inequality $F_{max} = \frac{Q_{worst} - s_e \cdot S_b}{S_q} \approx \frac{Q_{worst}}{S_q} \ge fpp_e = 0.5^k$



$$\boxed{log_{0.5}\left(\frac{Q_{worst} - s_e \cdot S_b}{S_q}\right)} \le k \le \boxed{\frac{M'}{1.44 \cdot C}} \qquad (10)$$

Performance requirement limitation

Memory space limitation