



mpi

Network-Centric
Information Systems
NetCInS



***KLEE*: A Framework for Distributed Top-k Query Algorithms**

Sebastian Michel

Max-Planck Institute for Informatics
Saarbrücken, Germany
smichel@mpi-inf.mpg.de

Peter Triantafillou

RACTI / Univ. of Patras
Rio, Greece
peter@ceid.upatras.gr

Gerhard Weikum

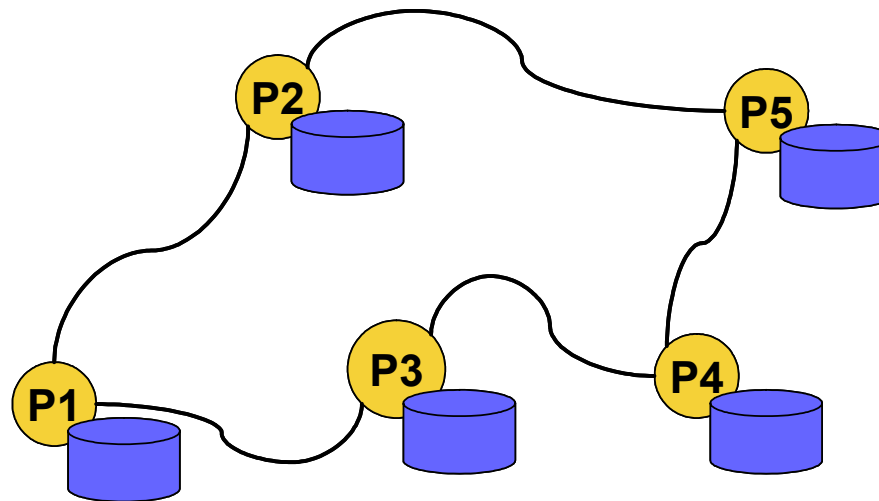
Max-Planck Institute for Informatics
Saarbrücken, Germany
weikum@mpi-inf.mpg.de

Overview

- **Problem Statement**
- **Related Work**
- **KLEE**
- **The Histogram Bloom Structure**
- **Candidate Filtering**
- **Evaluation**
- **Conclusion / Future Work**

Computational Model

- **Distributed aggregation queries:**
Query with m terms with index lists spread across m peers $P1 \dots Pm$

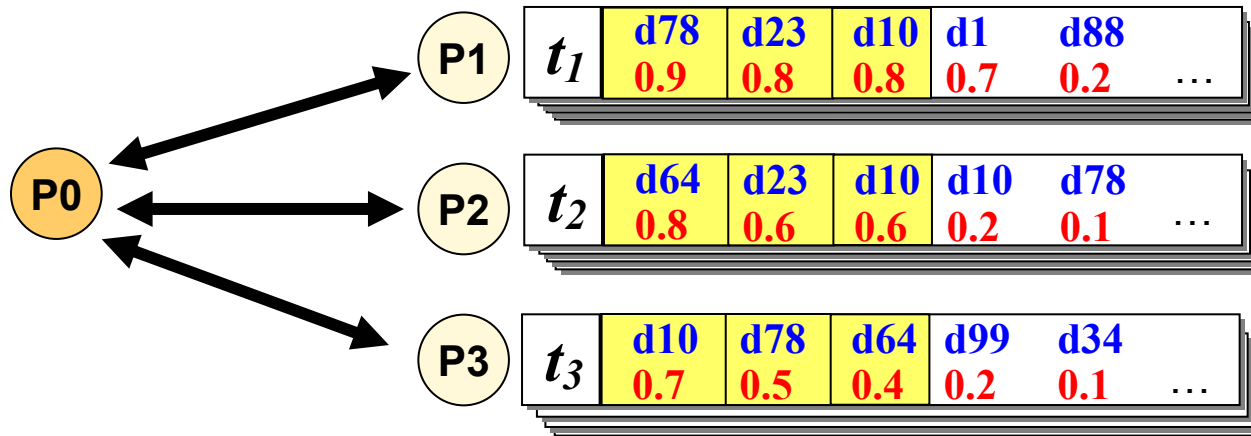


Applications:

- Internet traffic monitoring
- Sensor networks
- P2P Web search

Problem Statement

Query initiator P0 serves as per-query coordinator



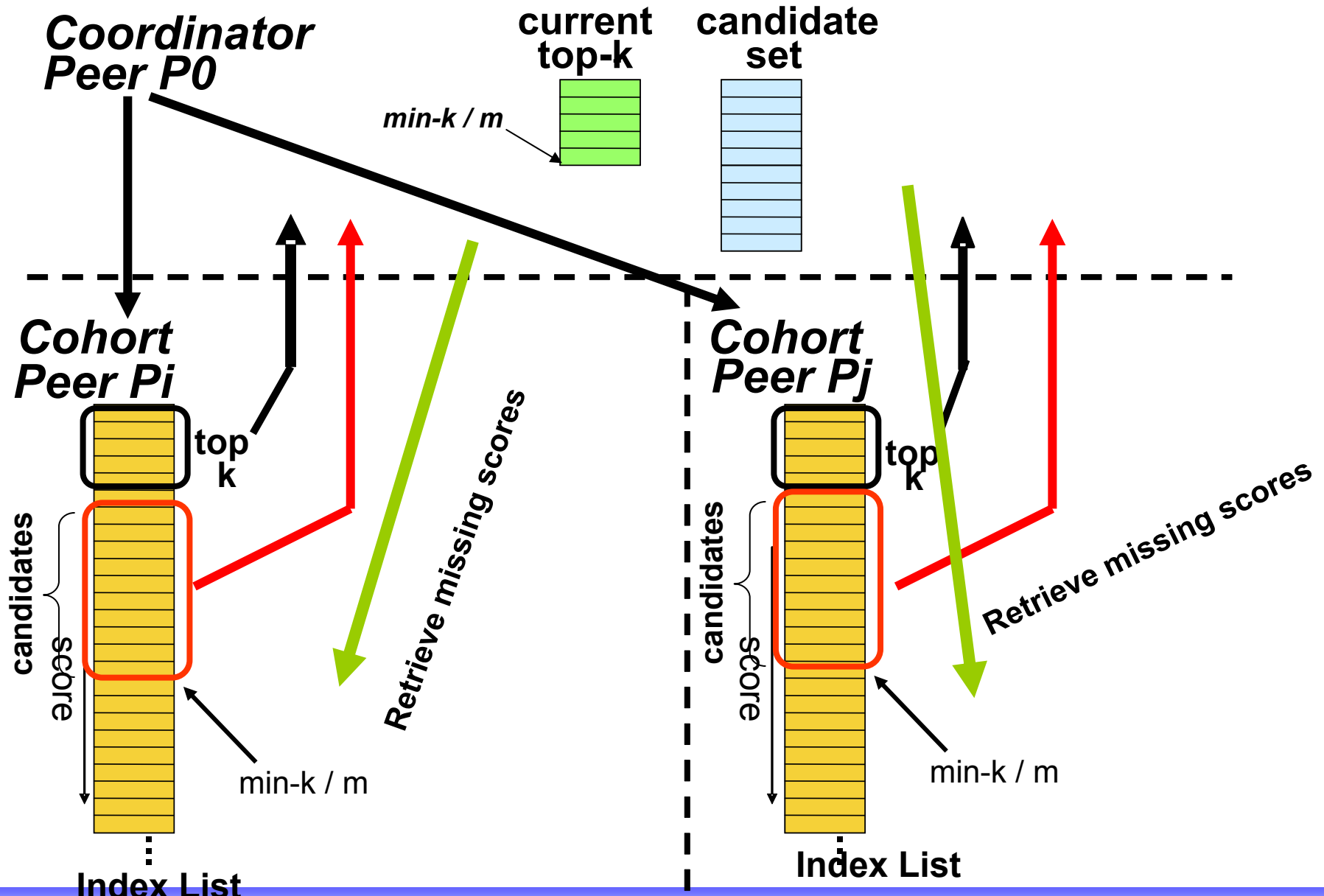
- Consider
 - network consumption
 - per peer load
 - latency (query response time)
 - network
 - I/O
 - processing

Related Work

Existing Methods:

- **Distributed NRA/TA:** Extend NRA/TA (Fagin et al. '99/'03, Guntzer et al. '01, Nepal et al. '99) with batched access
 - **TPUT (Cao/Wang 2004):**
 - 1) fetch k best entries (d, s_j) from each of $P_1 \dots P_m$ and aggregate $(\sum_{j=1..m} s_j(d))$ at P_0
 - 2) ask each of $P_1 \dots P_m$ for all entries with $s_j > \min-k / m$ and aggregate results at P_0
 - 3) fetch missing scores for all candidates by random lookups at $P_1 \dots P_m$
- + DNRA aims to minimize per-peer work
- DTA/DNRA incur many messages
- + TPUT guarantees fixed number of message rounds
- TPUT incurs high per-peer load and net BW

TPUT



KLEE: Key Ideas

- if $\text{min}k / m$ is small TPUT retrieves a lot of data in Phase 2
 - ➔ high network traffic
- random accesses
 - ➔ high per-peer load

KLEE:

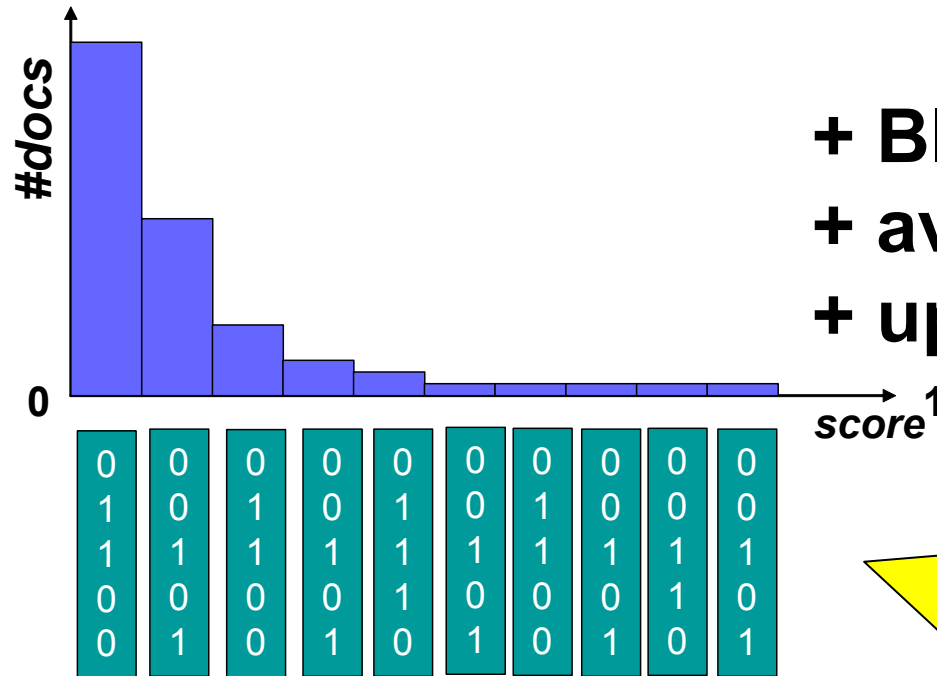
- Different philosophy: **approximate** answers!
- **Efficiency:**
 - Reduces (docId, score)-pair transfers
 - no random accesses at each peer
- Two pillars:
 - The **HistogramBlooms** structure
 - The **Candidate List Filter** structure

The KLEE Algorithms

- **KLEE 3 or 4 steps:**
 1. **Exploration Step:** ... to get a better approximation of min-k score threshold
 2. **Optimization Step:**
 - decide: 3 or 4 steps ?
 3. **Candidate Filtering:** ... a docID is a good candidate if high-scored in many peers.
 4. **Candidate Retrieval:** get all good docID candidates.

Histogram Bloom Structure

- Each peer pre-computes for each index list:
an equi-width histogram



- + Bloom filter for each cell
- + average score per cell
- + upper/lower score

**“increase” the mink / m
threshold**

Bloom Filter

- bit array of size m
- k hash functions

$$h_i: docid_space \rightarrow \{1, \dots, m\}$$

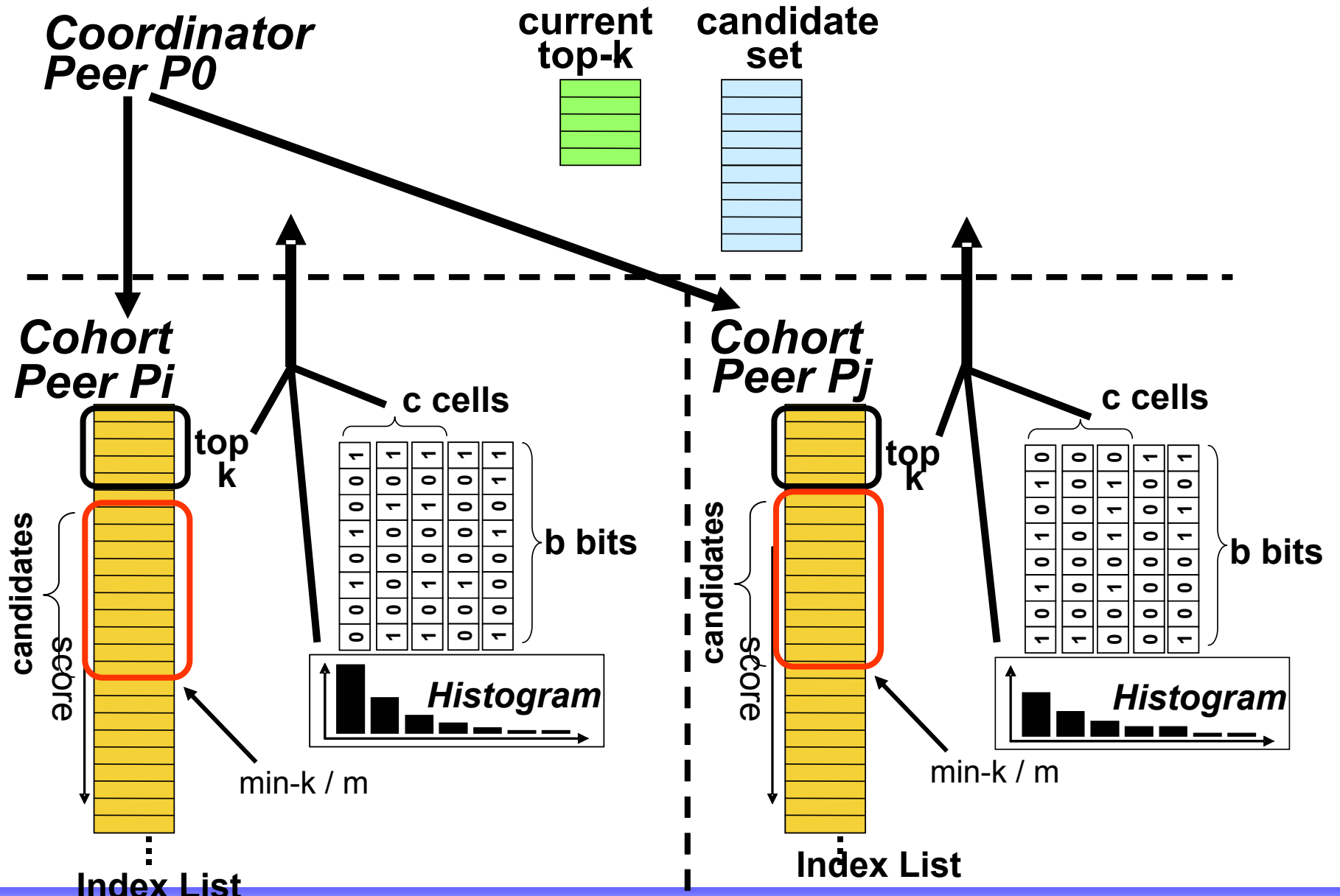
- insert n docs by hashing the ids and settings the corresponding bits

- **Membership Queries:**

- document is in the Bloom Filter if the corresponding bits are set

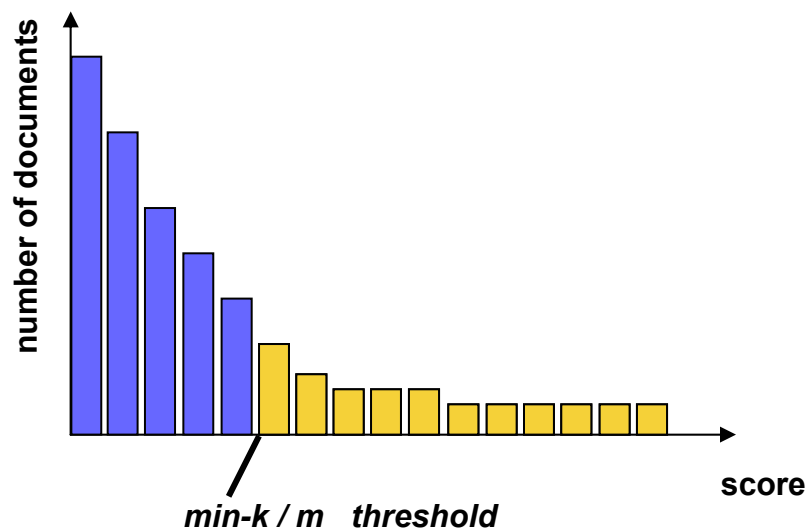
- probability of false positives (*pfp*) $pfp = (1 - e^{-kn/m})^k$
- tradeoff accuracy vs. efficiency

Exploration and Candidate Retrieval



Candidate List Filter Matrix

- **Goal: filter out unpromising candidate documents in step 2**
- **estimate the max number of docs that are above the $\text{min-}k / m$ threshold**

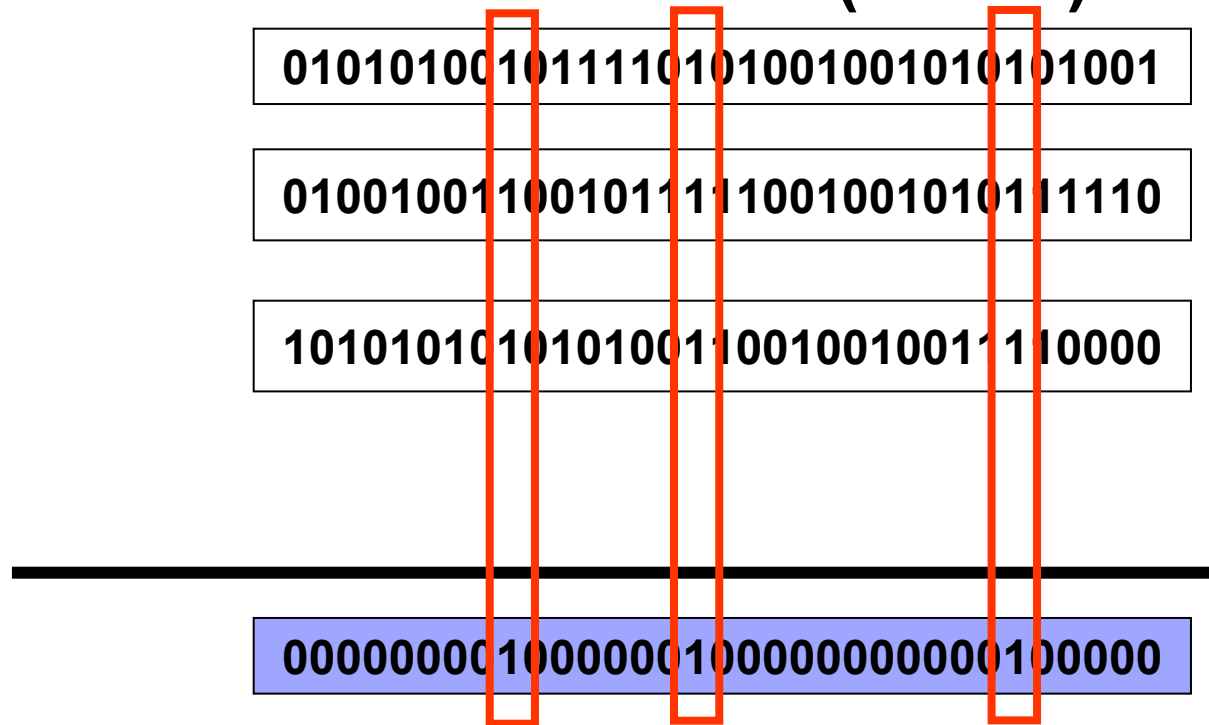


- **send this number and the threshold to the cohort peers**

Candidate List Filter Matrix (2)

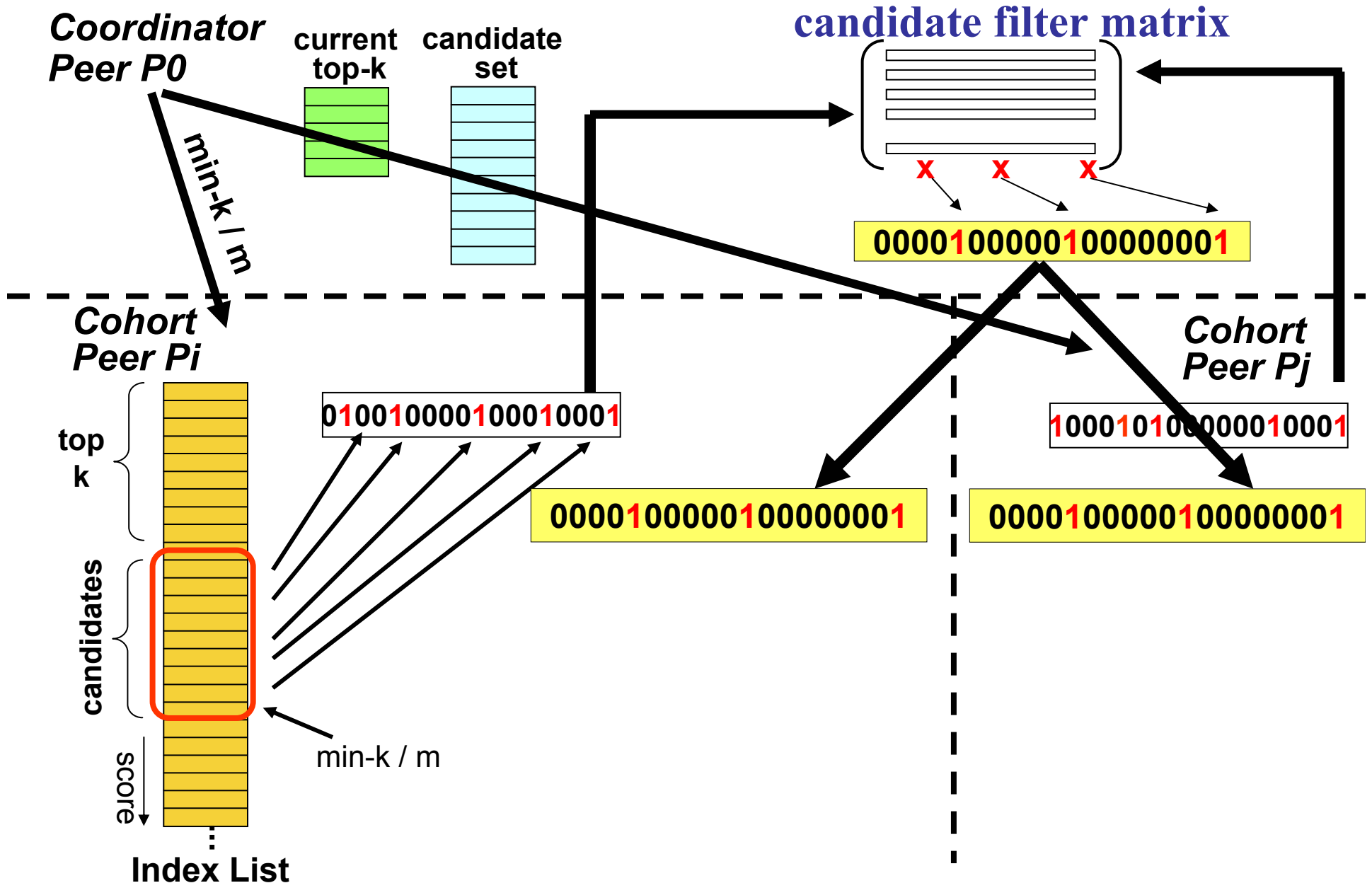
- Each cohort returns a Bloom Filter that “contains” all docs above the $\text{min}k / m$ threshold

→ Candidate List Filter Matrix (CLFM)

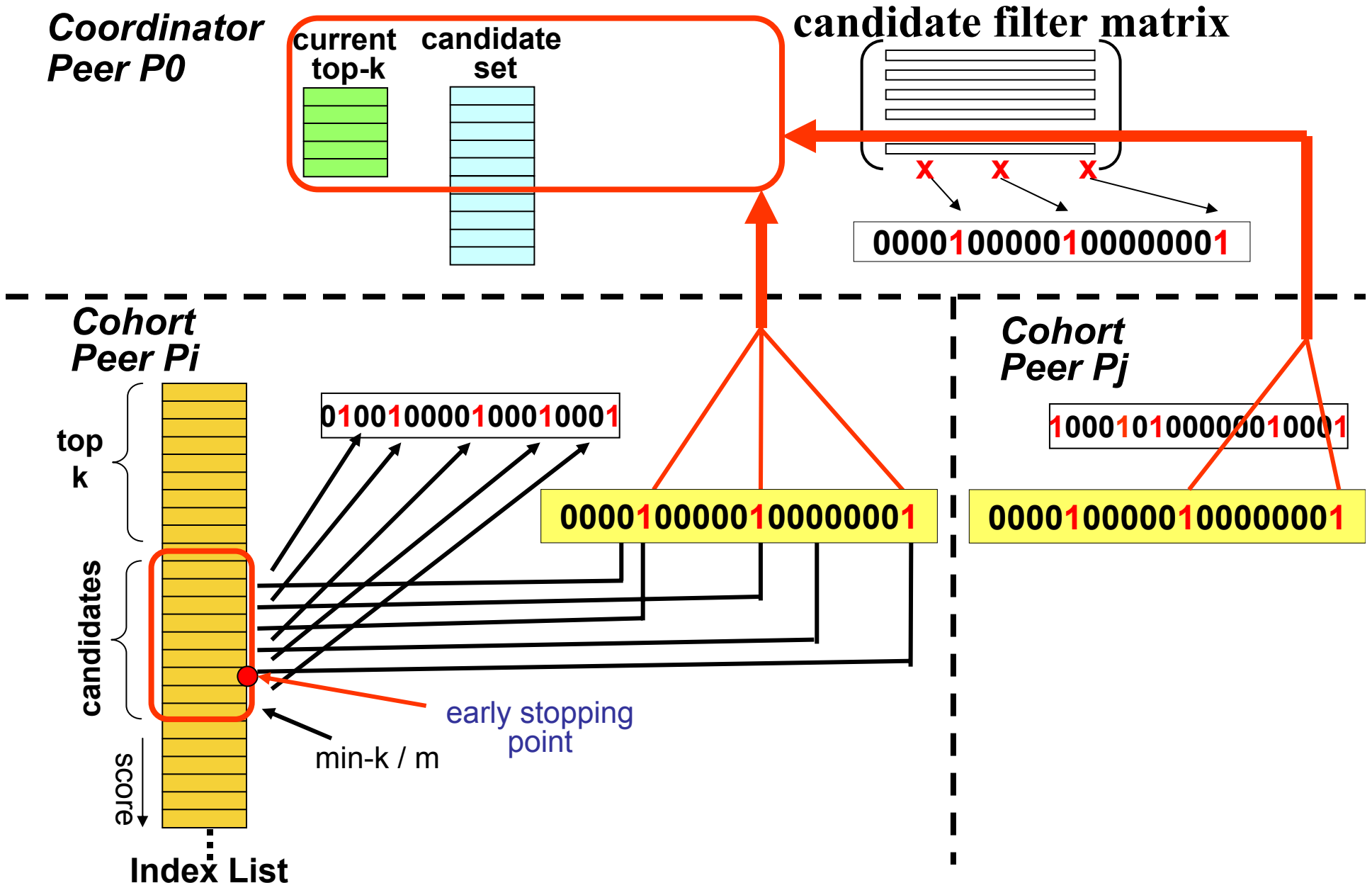


Select all columns with at least R bits set

KLEE- Candidate Set Reduction

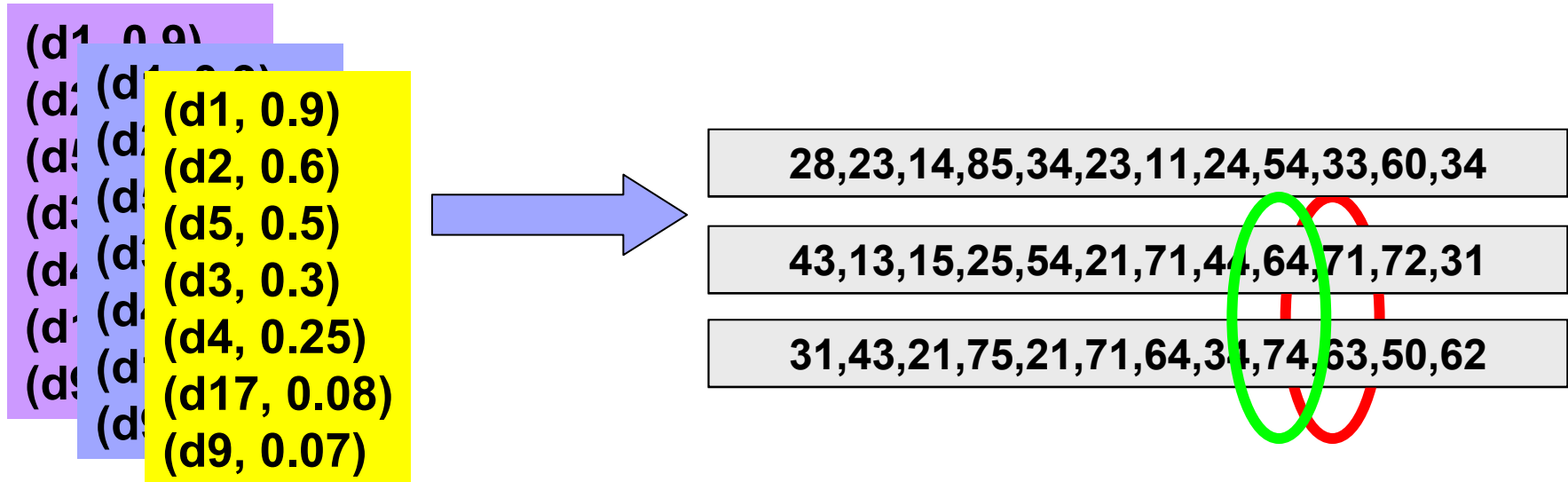


KLEE – Candidate Retrieval



Enhanced Filtering

- BF representation can be improved ...

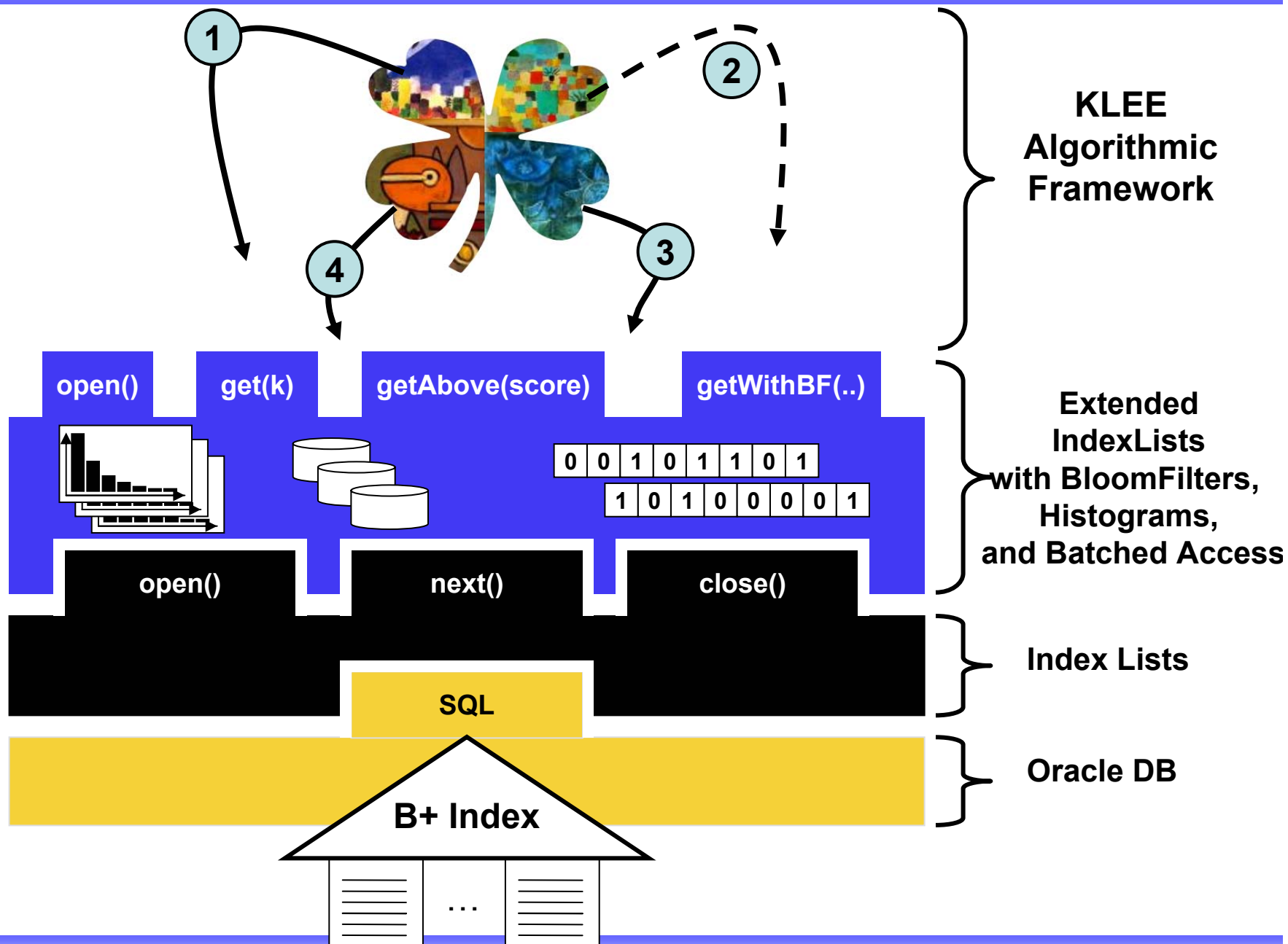


d1, d2, and d5 are promising documents but
e.g. $s_1-s_5 = 0.4$!

- Send byte-array with cell-numbers instead of bits
- Select „columns“ with

Sum over upper-bounds $>$ min-k

Architecture/Testbed



Evaluation: Benchmarks

- **GOV**: TREC .GOV collection + 50 TREC-2003 Web queries, e.g. *juvenile delinquency*
- **XGOV**: TREC .GOV collection + 50 manually expanded queries, e.g. *juvenile delinquency youth minor crime law jurisdiction offense prevention*
- **IMDB**: Movie Database, queries like
 - actor = John Wayne; genre =western
- **Synthetic Distribution** (Zipf, different skewness): GOV collection but with synthetic scores
- **Synthetic Distribution + Synthetic Correlation**: 10 index lists

Evaluation: Metrics

- **Relative recall w.r.t. to the actual results**
- **Score error**
- **Bandwidth consumption**
- **Rank distance**
- **Number of RA and number of SA**
- **Query response time**
 - network cost (150ms RTT,
800Kb/s data transfer rate)
 - local I/O cost (8ms rotation latency
+ 8MB/s transfer delay)
 - processing cost

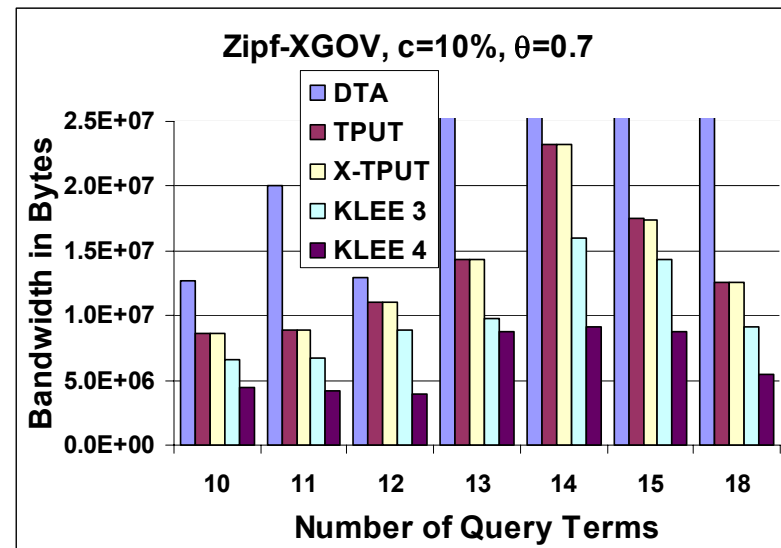
Evaluated Algorithms

- **DTA:**
 - **batched distributed threshold algorithm, batch size k .**
 - **TPUT**
 - **X-TPUT:**
 - **approximate TPUT. No random accesses.**
 - **KLEE-3**
 - **KLEE-4**
- C = 10% of the score mass**

Synthetic Score Benchmarks

Zipf-GOV c=10%	Total # of Bytes	Total Time in ms	Average Recall
DTA	17,752,769	3,532,180	1
TPUT	53,494,903	576,713	1
X-TPUT	53,011,252	404,991	0.99
KLEE 3	49,861,342	367,931	0.97
KLEE 4	25,057,920	160,585	0.94

Zipf-XGOV c=10%	Total # of Bytes	Total Time in ms	Average Recall
DTA	617,009,260	39,582,682	1
TPUT	377,928,880	1,599,581	1
X-TPUT	377,097,644	1,521,220	0.98
KLEE 3	287,294,812	1,189,891	0.91
KLEE 4	165,077,807	375,077	0.92

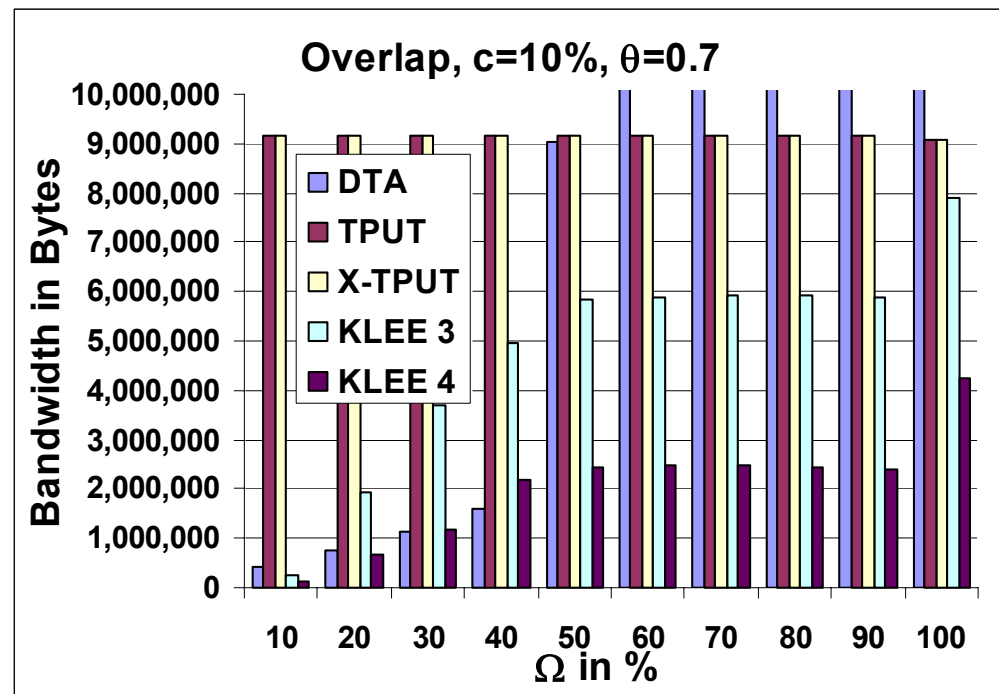
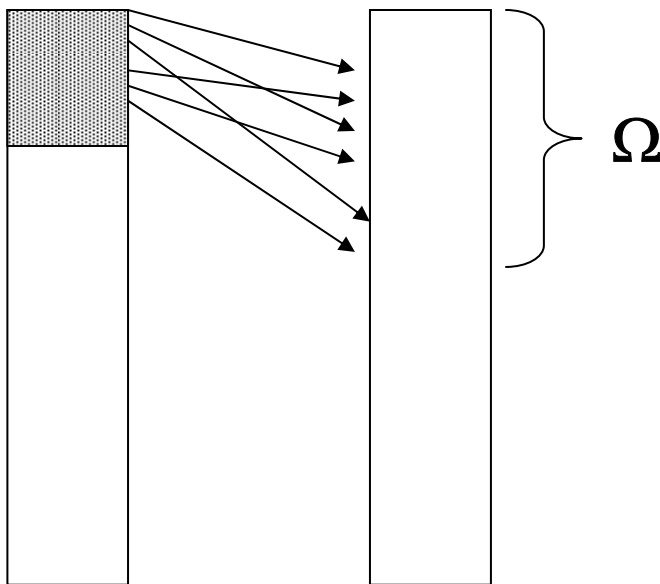


$\theta = 0.7$

Synthetic Correlation Benchmark

Overlap+Zipf c=10% $\Omega = 30\%$	Total # of Bytes	Total Time in ms	Average Recall
DTA	1,146,32	157,420	1
TPUT	9,150,904	29,270	1
X-TPUT	9,150,904	28,335	1
KLEE 3	3,678,780	12,971	0.92
KLEE 4	1,192,704	6,546	0.91

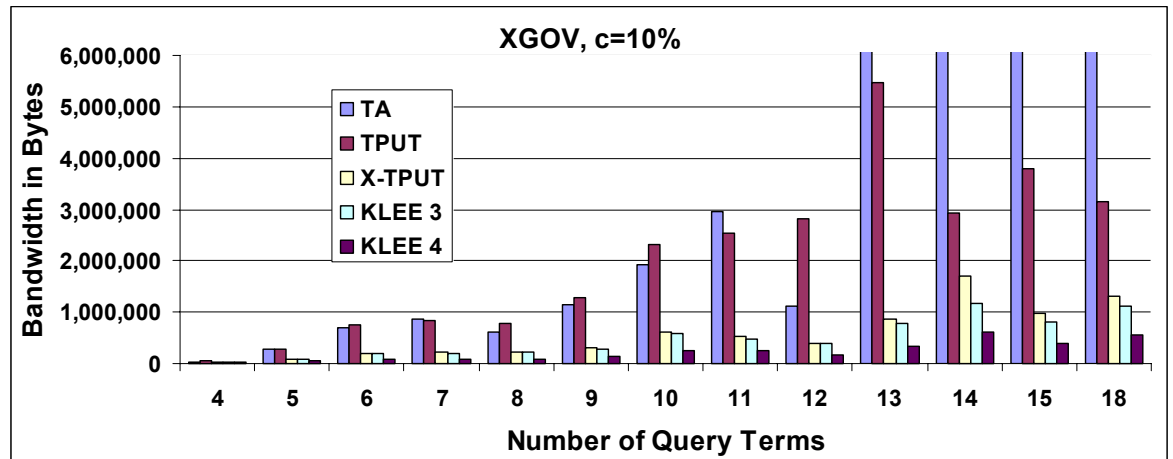
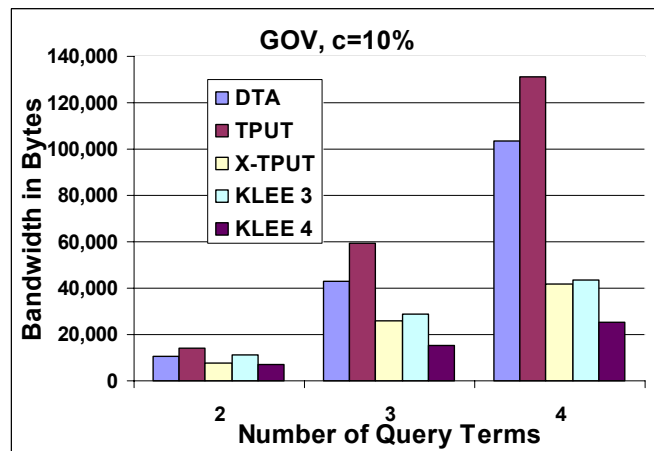
randomly insert top k documents from list i
in the top Ω documents of list j



GOV / XGOV

GOV c=10%	Total # of Bytes	Total Time in ms	Average Recall
DTA	1,172,446	190,259	1
TPUT	1,505,290	185,049	1
X-TPUT	597,991	31,432	0.89
KLEE 3	722,664	28,319	0.9
KLEE 4	440,868	39,564	0.9

XGOV c=10%	Total # of Bytes	Total Time in ms	Average Recall
DTA	92,587,264	3,740,677	1
TPUT	70,044,884	2,346,882	1
X-TPUT	19,236,084	96,153	0.91
KLEE 3	16,690,912	88,271	0.83
KLEE 4	7,920,774	56,609	0.79



Conclusion / Future Work

- **Conclusion**
 - **KLEE: approximate top-k algorithms for wide-area networks**
 - **significant performance benefits can be enjoyed, at only small penalties in result quality**
 - **flexible framework for top-k algorithms, allowing for trading-off**
 - **efficiency versus result quality and**
 - **bandwidth savings versus the number of communication phases.**
 - **various fine-tuning parameters**
- **Future Work**
 - **Reasoning about parameter values**
 - **Consider “moving” coordinator**

Thanks for your attention!

Questions?

Comments?

