Read as Needed: Building WiSER, a Flash-Optimized Search Engine

He Jun[1], Wu Kan[1], Kannan Sudarsun[2], Arpaci-Dusseau Andrea[1],
Arpaci-Dusseau Remzi[1]

[1]Department of Computer Sciences, University of Wisconsin - Madison

[2]Department of Computer Sciences, Rutgers University

FAST'20

Summaries

➢ What?

- Based on read as needed rather than cache to build an Elasticsearch search engine “WiSER”, which is optimized for small RAM with high throughput SSD.

➢ Why?

- RAM is far more expensive than SSD, and only use RAM may waste the bandwidth (~100GB/s).
- The total capacity of RAM is much smaller than SSD on a single machine.

➢ How?

- Reduce read amplification.
- Hide I/O latency.
- Use large request to improve device efficiency.

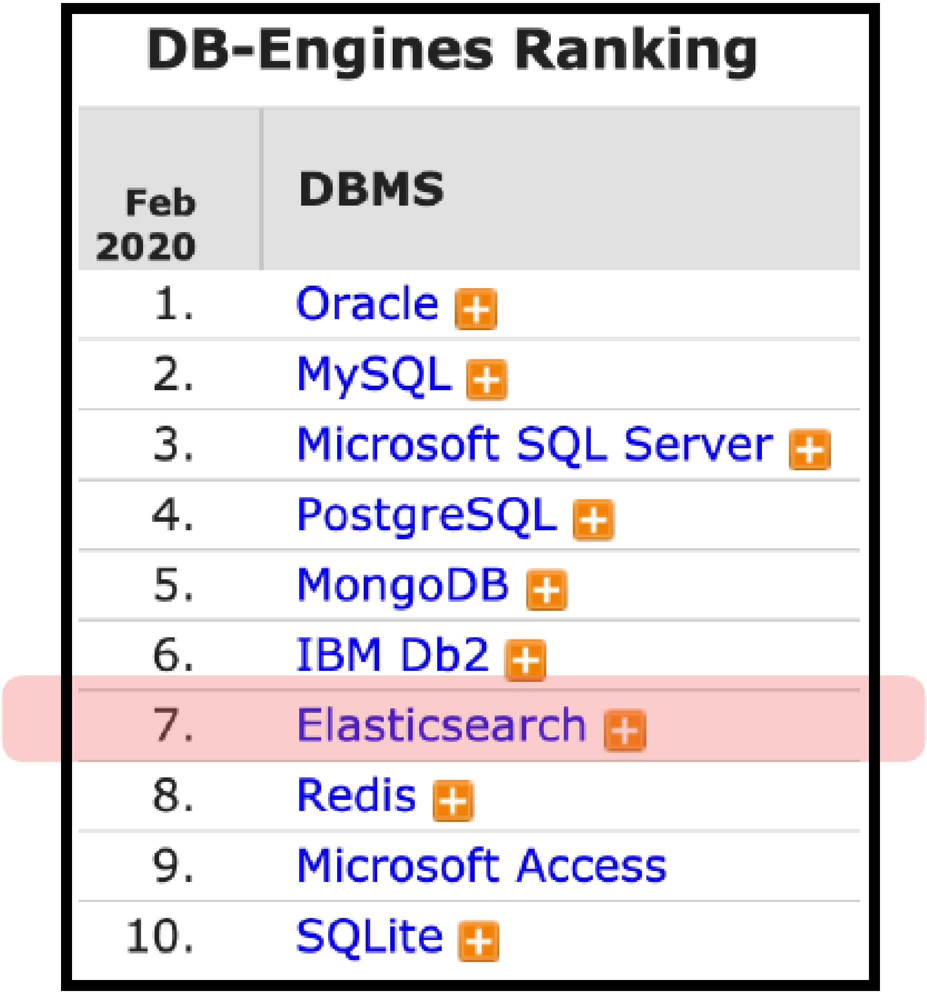
Background

SSD or RAM

Technical index	SSD	RAM
Performance	Sequential read: 3.5GB/s(PCIe3.0) 4.5GB/s(PCIe4.0)	Single Channel: DDR4 2400 19.2GB/s DDR4 3200 25.6GB/s
	Random read: 500000 IOPS	
	Limited by the number of PCIE buses	Increases linearly with the number of channels
latency	<20us	<100ns
Price	MLC: ~0.3$/GB TLC:~0.1$/GB	~5$

Background

- **Elasticsearch search engine**
  - Provides a full-text search engine with distributed multi-user capabilities, based on a RESTful web interface.
  - Requires： Low latency; High throughput; High scalability
  - Widely used in Wikipedia, GitHub, Uber…



DB-Engines Ranking

Feb 2020	DBMS
1.	Oracle
2.	MySQL
3.	Microsoft SQL Server
4.	PostgreSQL
5.	MongoDB
6.	IBM Db2
7.	Elasticsearch
8.	Redis
9.	Microsoft Access
10.	SQLite

Motivation

➢ Current status:

- Key-value stores, File systems are optimized for SSD, but Search engines are overlooked.
- RAM is the bottleneck of search engines.
- The growth rate of data far exceeds the growth rate of memory, and the growth of memory is subject to the double constraints of computer structure and cost.

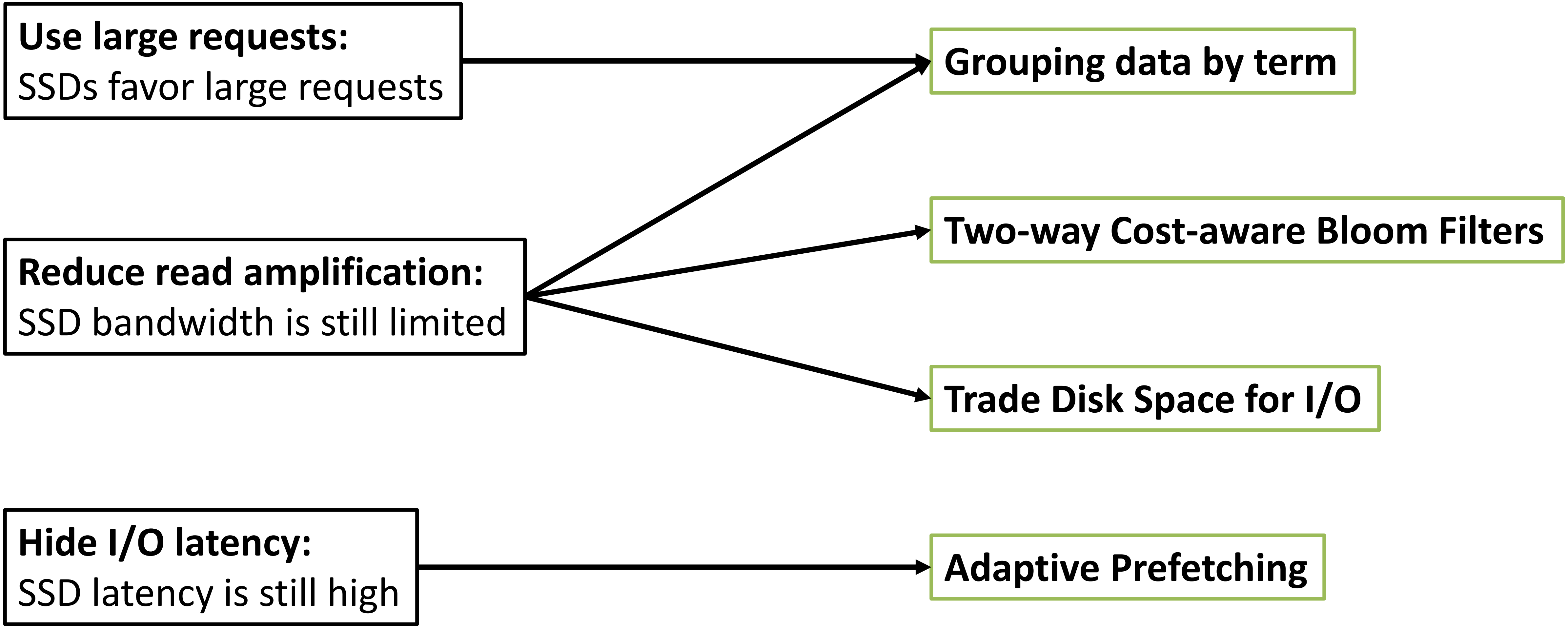
➢ The main Problem:

- **Can search engines perform well with a small memory and a fast SSD?**

Motivation

- **Observations**
  - SSDs have much higher read bandwidth and IOPS than ever before.
  - In most cases, the throughput is higher than the upper limit of network bandwidth and computing power.
- **The answer to the main problem:**
  - If read data from SSDs more efficiently (Read as Needed), built an engine with small memory and fast SSDs have chance to outperform other engine with entire dataset in memory.

How to Read as Needed?



Use large requests:
SSDs favor large requests

Reduce read amplification:
SSD bandwidth is still limited

Hide I/O latency:
SSD latency is still high

Grouping data by term

Two-way Cost-aware Bloom Filters

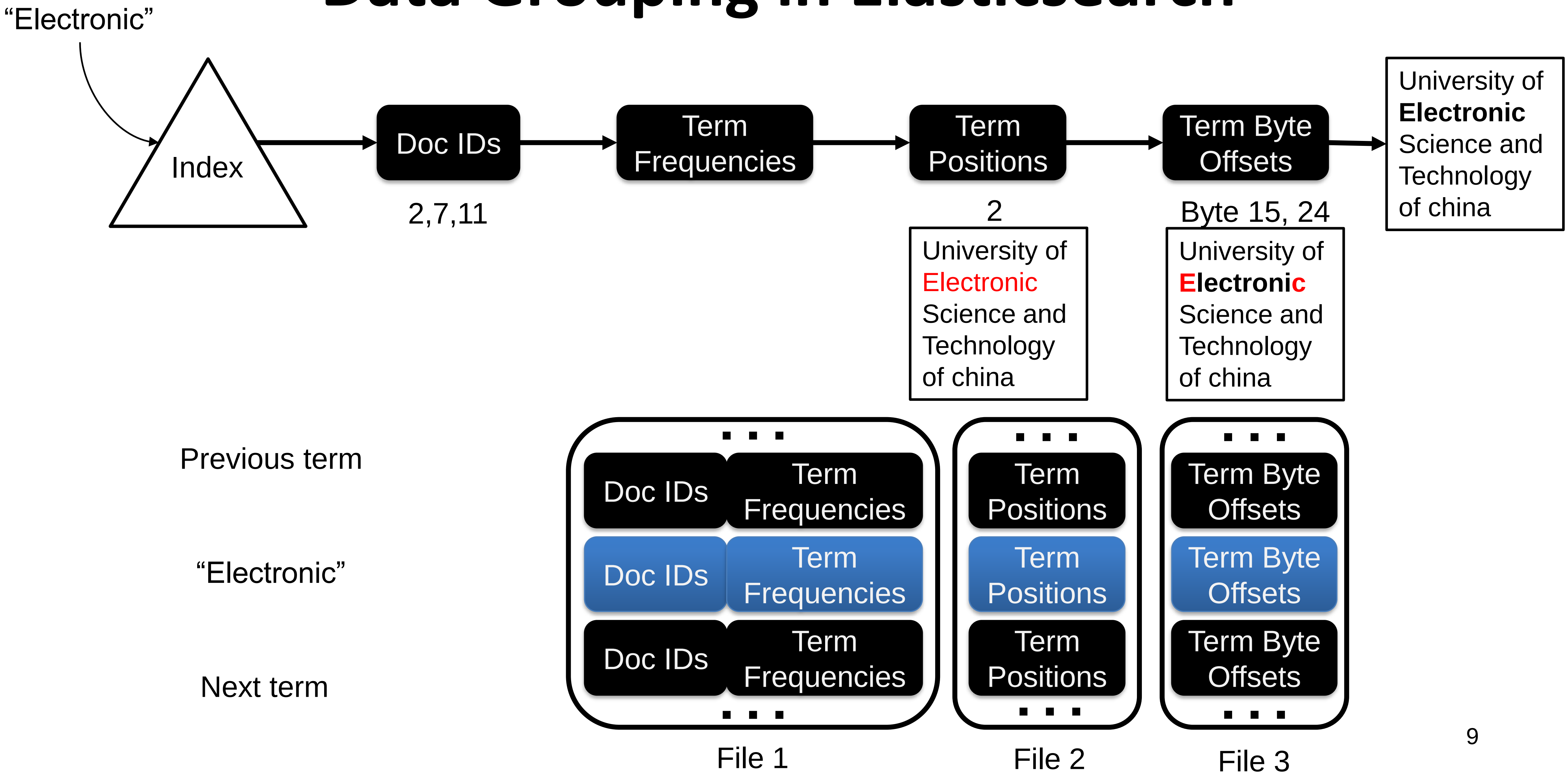
Trade Disk Space for I/O

Adaptive Prefetching

Techniques

- ➢ Cross-stage Data Grouping
- ➢ Two-way Cost-aware Bloom Filters
- ➢ Adaptive Prefetching
- ➢ Trade Disk Space for I/O

Data Grouping in Elasticsearch



“Electronic”

Index

Doc IDs

2,7,11

Term Frequencies

Term Positions

2

University of **Electronic** Science and Technology of china

Term Byte Offsets

Byte 15, 24

University of **Electronic** Science and Technology of china

University of **Electronic** Science and Technology of china

Previous term

“Electronic”

Next term

. . .

Doc IDs | Term Frequencies

Doc IDs | Term Frequencies

Doc IDs | Term Frequencies

. . .

File 1

. . .

Term Positions

Term Positions

Term Positions

. . .

File 2

. . .

Term Byte Offsets

Term Byte Offsets

Term Byte Offsets

. . .

File 3

Data Grouping in Elasticsearch

➢ The problem?

- Each search need three stage read operation.
- Three I/O count is needed for “Electronic”.
- Most transferred data is often wasted (Stage file 1, 2, 3 all need to load into memory).

➢ Observations

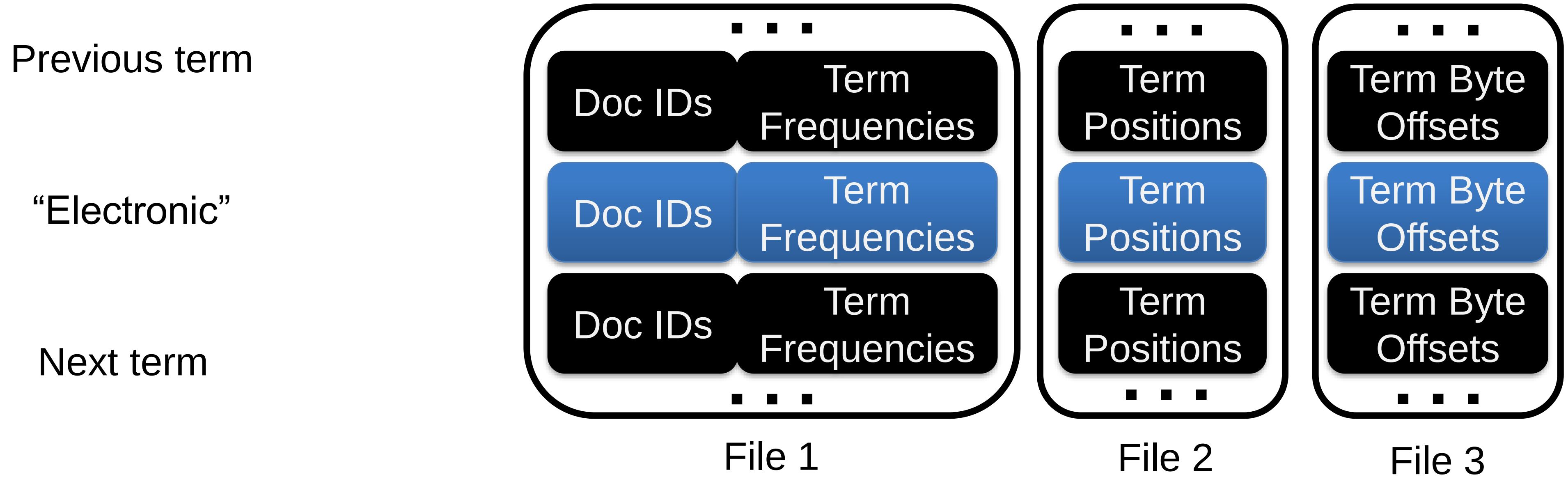
- Small term is commonly fit in 4KB (99% of Wikipedia terms can fit in 4KB).

➢ Solution

- **Group data by terms rather than by stage.**

Data Grouping in WiSER

New entry file



Previous term

“Electronic”

Next term

. . .

Doc IDs | Term Frequencies

Doc IDs | Term Frequencies

Doc IDs | Term Frequencies

. . .

File 1

. . .

Term Positions

Term Positions

Term Positions

. . .

File 2

. . .

Term Byte Offsets

Term Byte Offsets

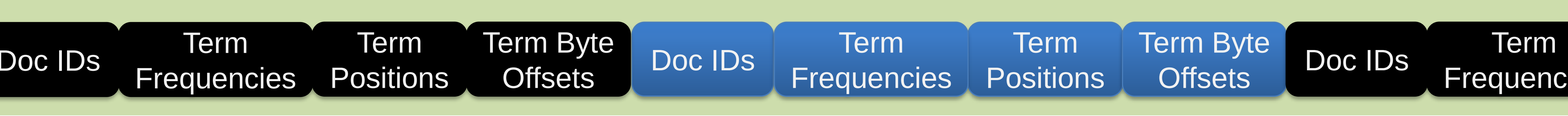
Term Byte Offsets

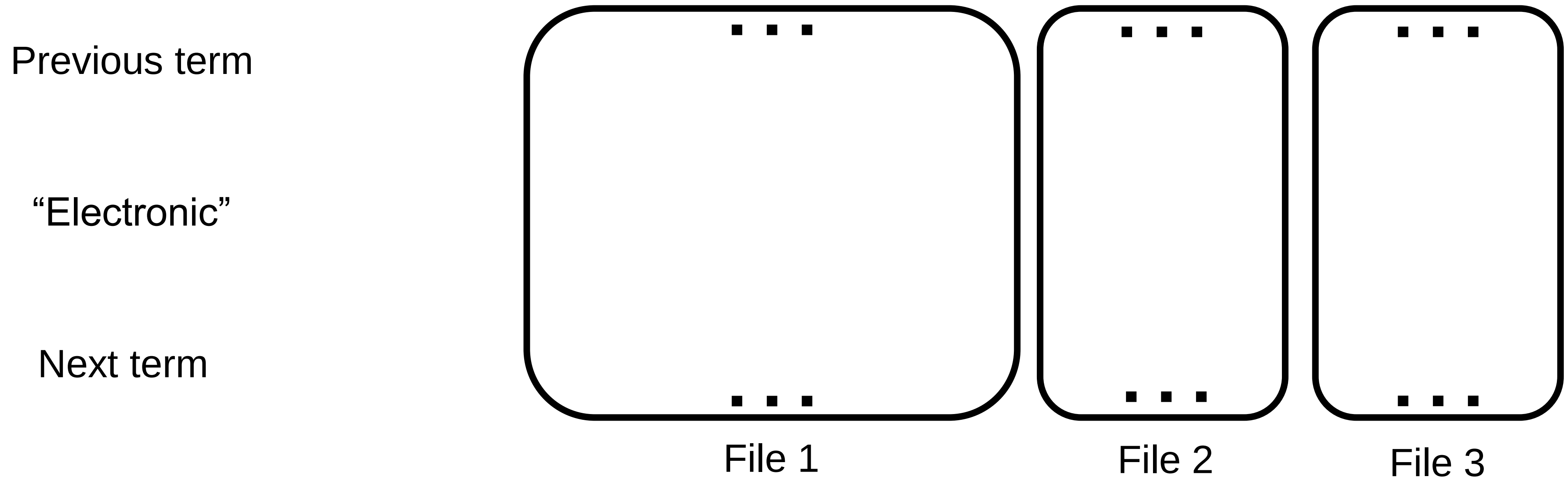
. . .

File 3

Data Grouping in WiSER

New entry file


Doc IDs
Term Frequencies
Term Positions
Term Byte Offsets
Doc IDs
Term Frequencies
Term Positions
Term Byte Offsets
Doc IDs
Term Frequenc

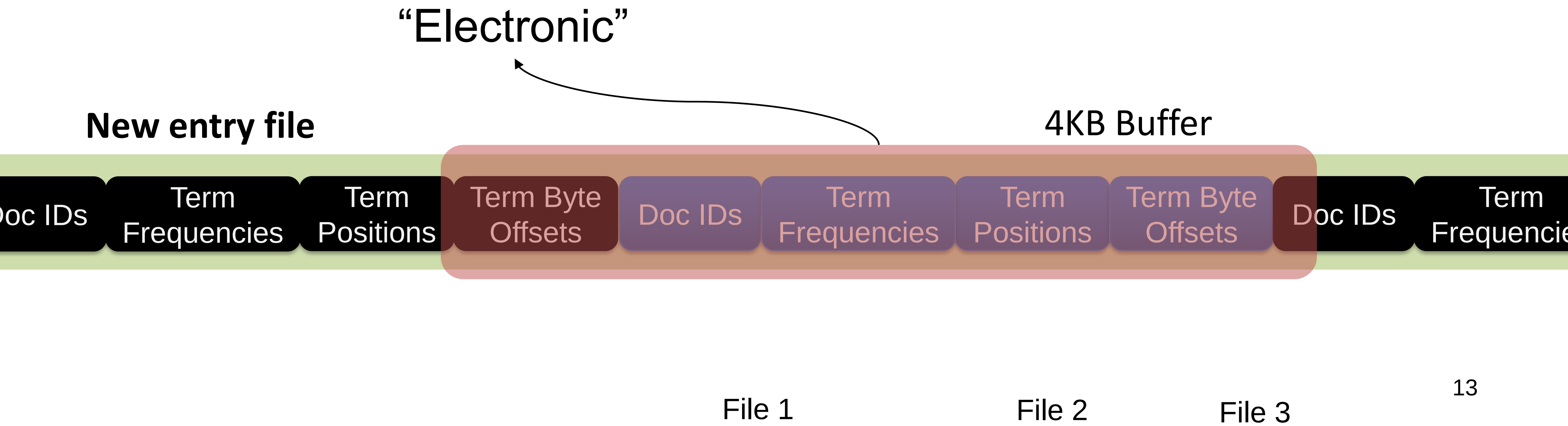

Previous term

“Electronic”

Next term

Data Grouping in WiSER

- **I/O cost for query “Electronic”?**
  - Total 1 I/O request is needed to get a 4KB buffer.



Techniques

- ➢ Cross-stage Data Grouping
- ➢ **Two-way Cost-aware Bloom Filters**
- ➢ Adaptive Prefetching
- ➢ Trade Disk Space for I/O

Two-way Cost-aware Bloom Filters

- **Bloom filter**
  - A data structure designed to tell you, rapidly and memory-efficiently, whether an element is present in a set.
  - It is a probabilistic data structure: it tells us that the element either definitely is not in the set or may be in the set.
- **Why basic Bloom Filters are not fit here?**
  - Phrase queries are very common in Elasticsearch workloads (which can provide higher accuracy), but require more data support, which will cause the Bloom Filter size to grow exponentially and lose effectiveness.
- **Solution**
  - **Use two bloom Filters to enable "Two-Way" filtering**

Two-way Cost-aware Bloom Filters

- **Check if “united states” is a phrase in a document**
  - Checking if “states” is in “united”->after.
  - Checking if “united” is in “states”->before.

Term	Bloom Filter (Before)	Bloom Filter (After)	Position
United	Our	States	2
States	United	President	3
United	Of	Airline	4
States	Airline	that	6

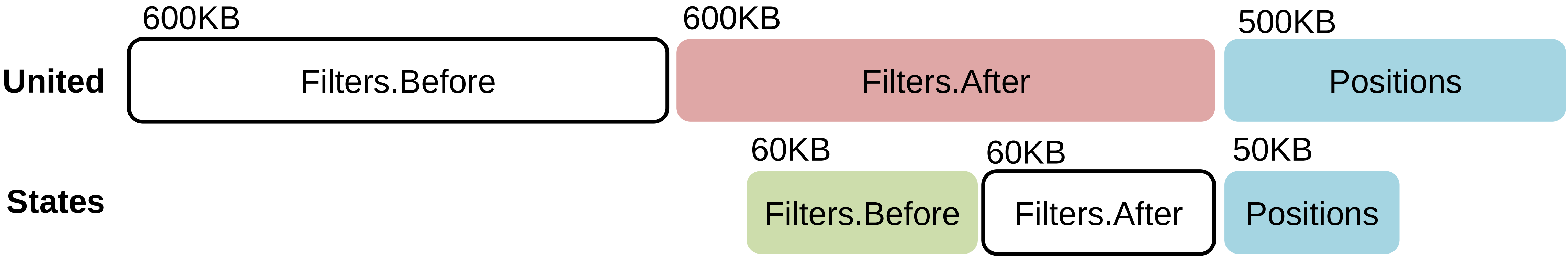
Our **United States** president Trump…

The spokesman of **united** airline **states** that …

Two-way Cost-aware Bloom Filters

➢ Cost-aware Analysis

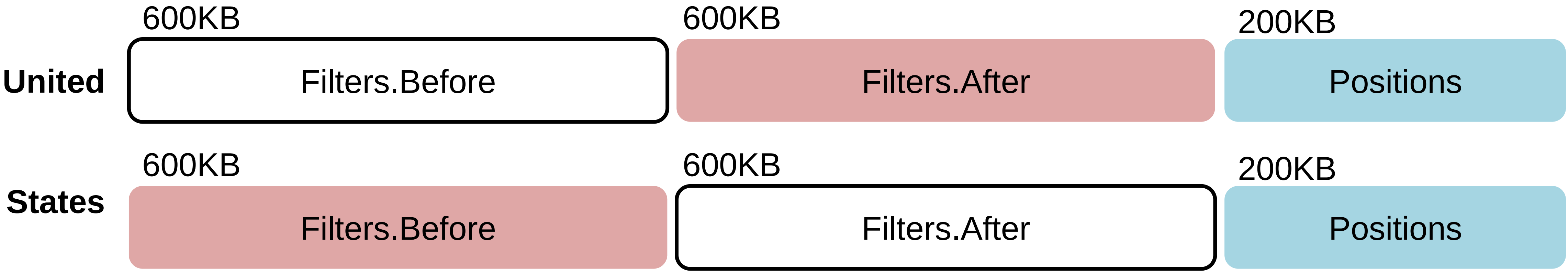
- Check if “states” is after “united”: check “states” Filters.Before



United: 600KB Filters.Before | 600KB Filters.After | 500KB Positions

States: 60KB Filters.Before | 60KB Filters.After | 50KB Positions

- Check if “states” is after “united”: not worth to use filter



United: 600KB Filters.Before | 600KB Filters.After | 200KB Positions

States: 600KB Filters.Before | 600KB Filters.After | 200KB Positions

Techniques

- ➢ Cross-stage Data Grouping
- ➢ Two-way Cost-aware Bloom Filters
- ➢ **Adaptive Prefetching**
- ➢ Trade Disk Space for I/O

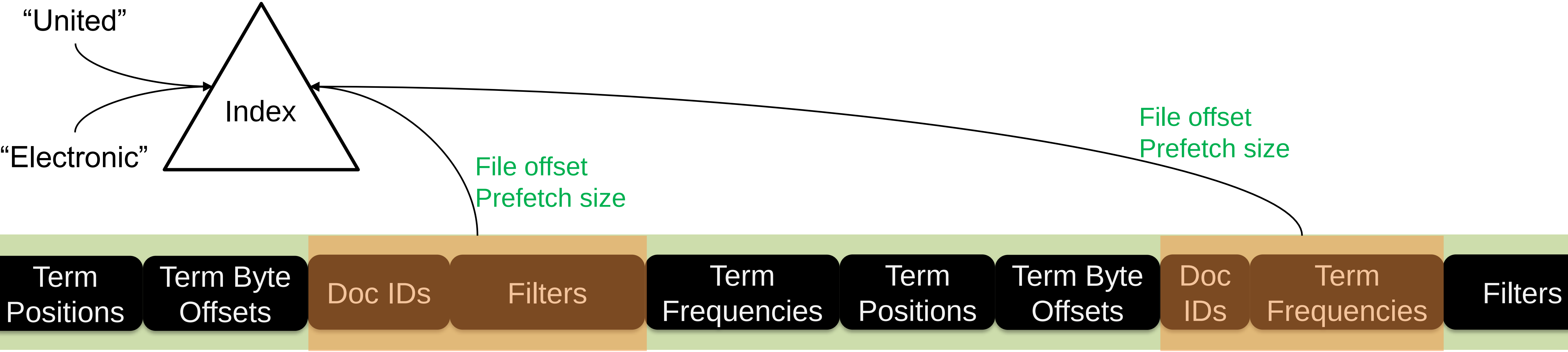
Adaptive Prefetching

➢ Elasticsearch Prefetch

- Depend on operating system (system read cache).

➢ Adaptive Prefetching

- Put commonly accessed data into a **Prefetch Zone** to help system prefetch content.
- Select “Prefetch Zone” data: metadata, skip list, document IDs, and term frequencies.

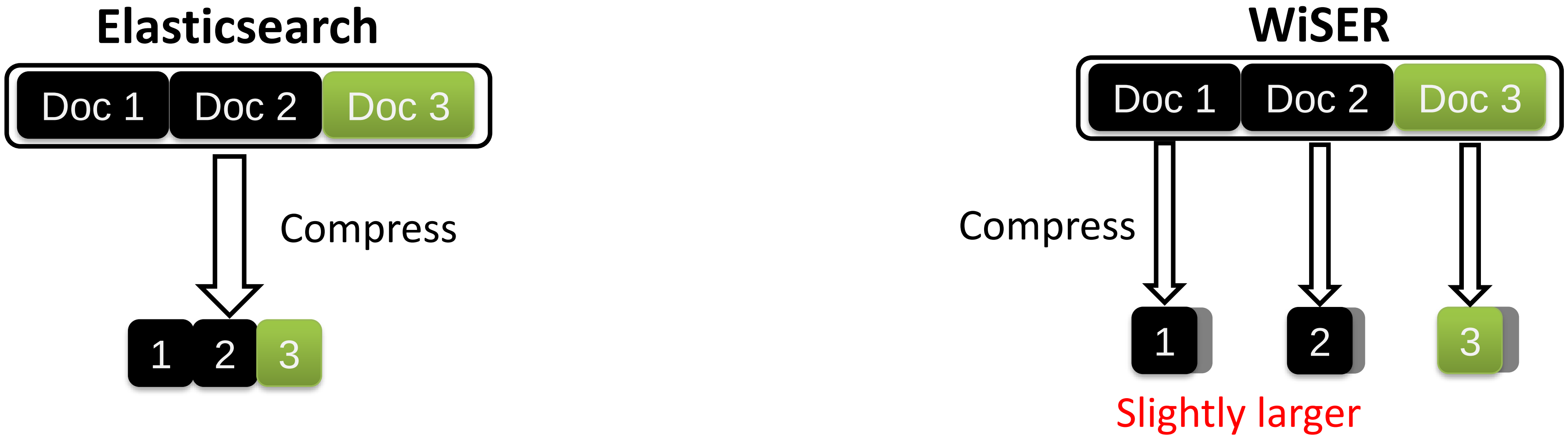


Techniques

- ➢ Cross-stage Data Grouping
- ➢ Two-way Cost-aware Bloom Filters
- ➢ Adaptive Prefetching
- ➢ **Trade Disk Space for I/O**

Trade Disk Space for I/O

- **Modify the data compression method**
  - Basic Elasticsearch compress many documents at same time, because LZ4. et. al. achieve better compression with larger input.
  - Compress each document individually to avoid reading and decompressing unnecessary documents with small space overhead.

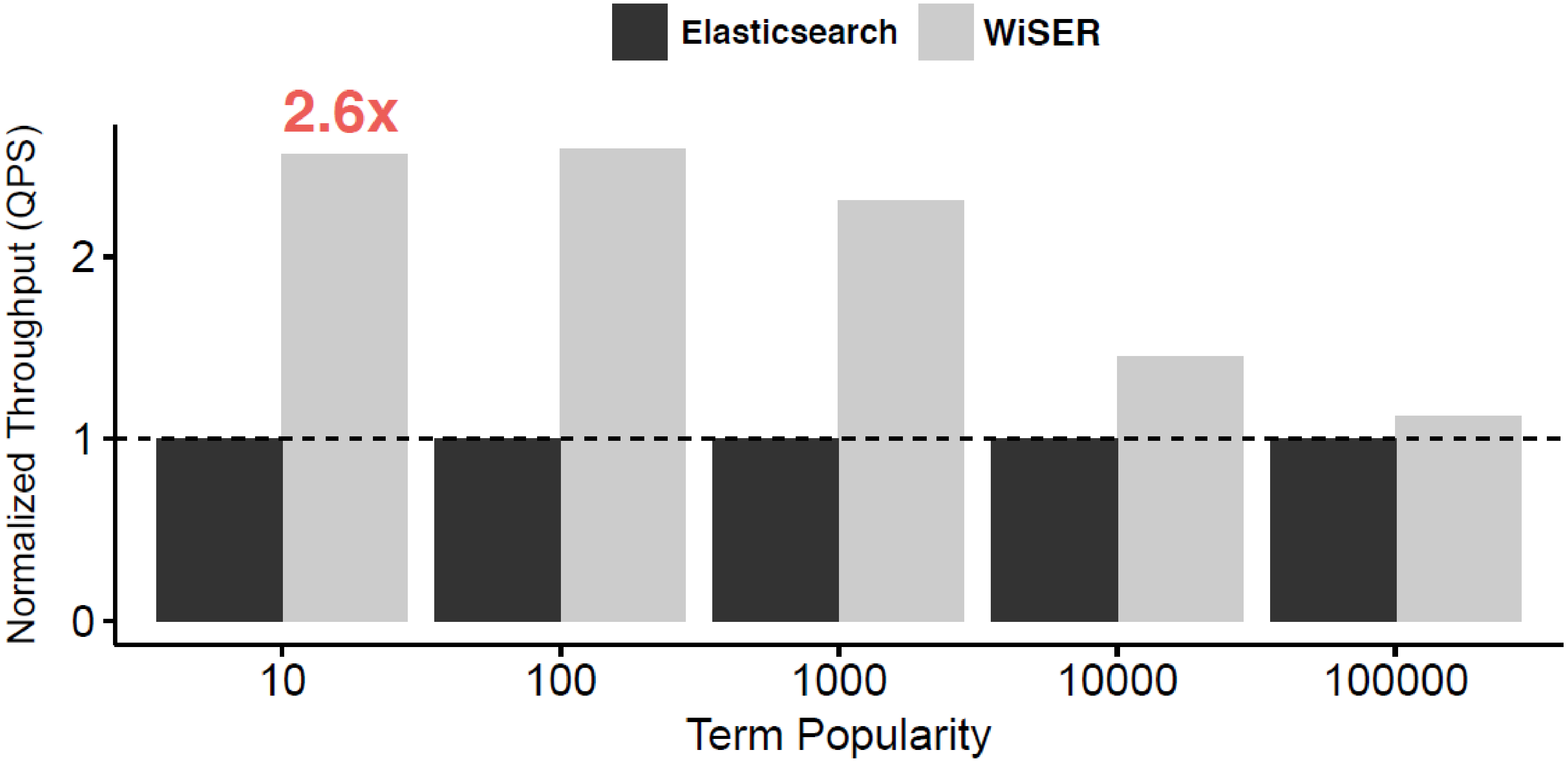


Evaluation

- **Dataset & Queries**
  - Wikipedia: 6 million documents, 6 million terms, 18GB.
  - single term queries, “and” queries, “phrase” queries, real queries.
  - vary term popularities in Wikipedia.
- **Machine**
  - NVMe SSD: peak read 2 GB/s, 200,000 IOPS
  - Linux container with 512MB RAM
- **Target**
  - **How effective is each search engine when the working sets cannot be fully loaded into memory?**

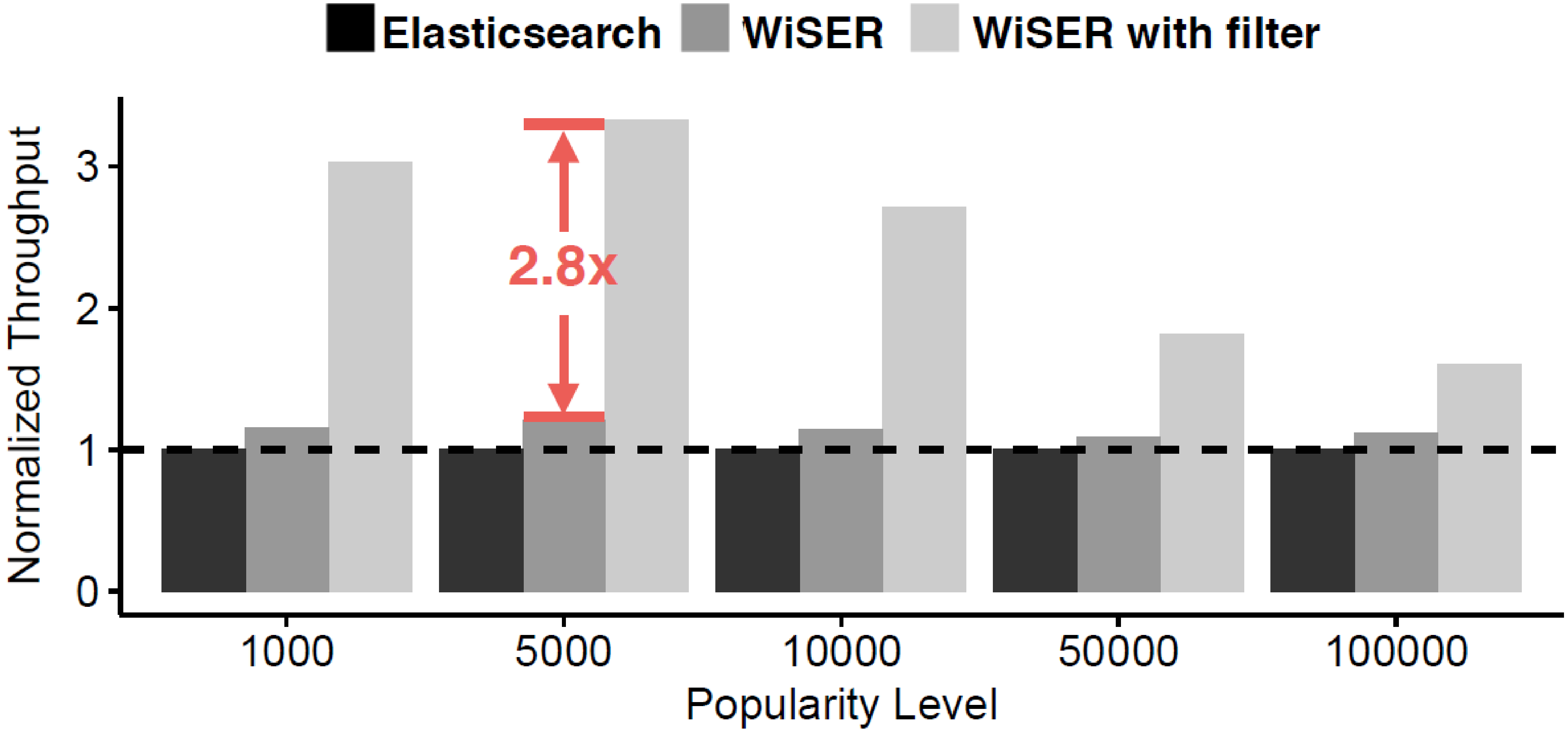
Evaluation

- Query throughput with “Cross-stage Data Grouping” method



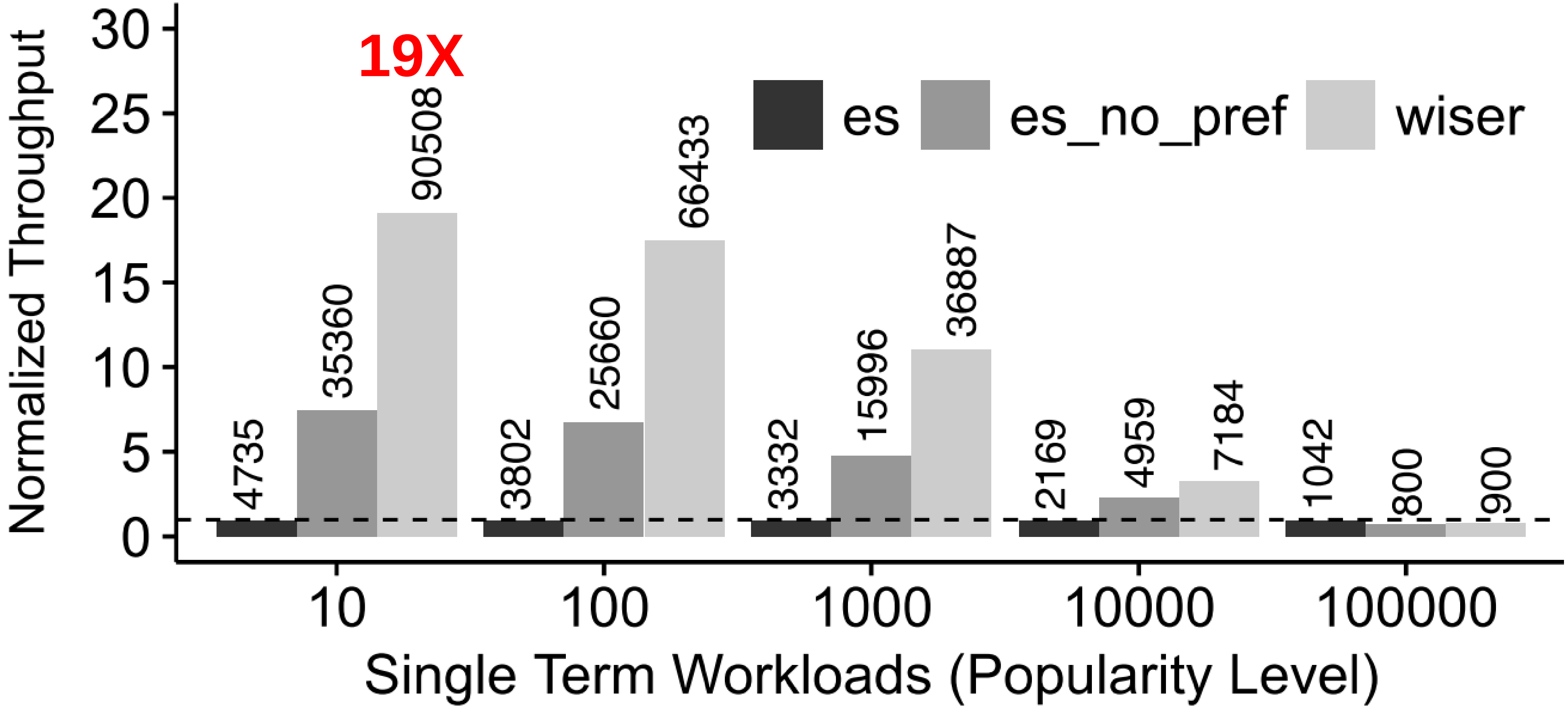
Evaluation

- Query throughput with “Two-way Cost-aware Bloom Filters” method



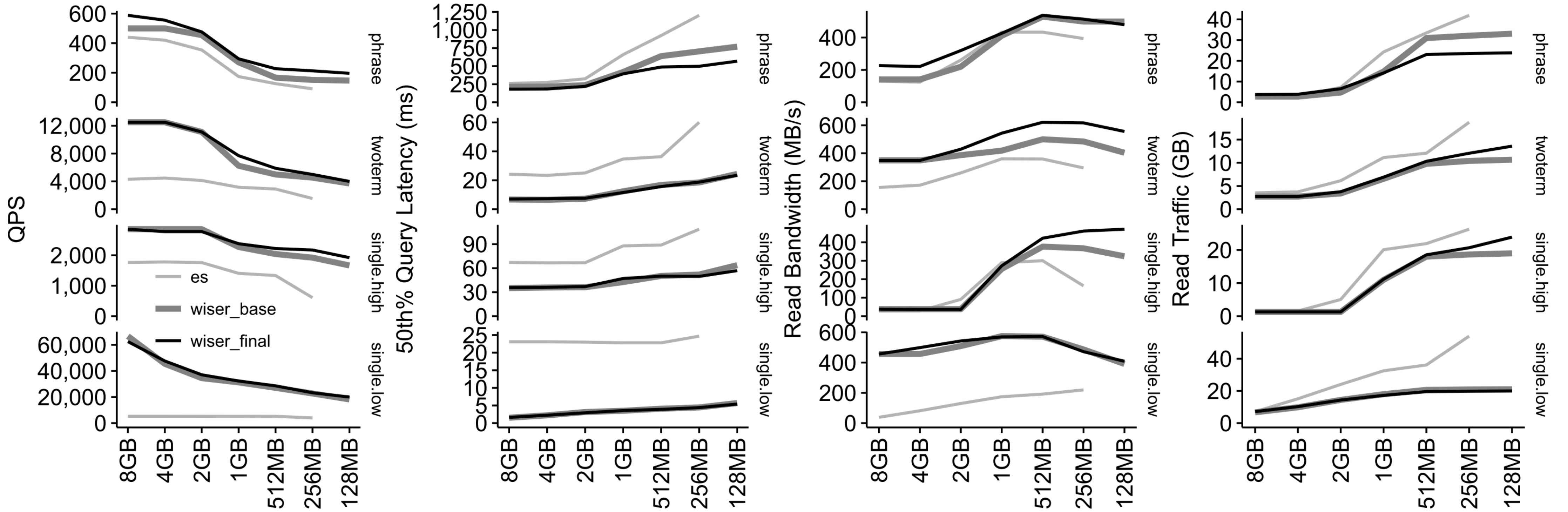
Evaluation

- **Query throughput with "Adaptive Prefetching & Trade Disk Space for I/O" method**



Evaluation

- **Overall performance with all techniques under different RAM size**
  - WiSER has significantly lower latency, higher throughput, and is more suitable for small memory



Conclusion

- **Introduce WiSER with 4 techniques:**
  - Cross-stage Data Grouping
  - Two-way Cost-aware Bloom Filters
  - Adaptive Prefetching
  - Trade Disk Space for I/O
- **Final thought**
  - Specially designed data structure, filter and special storage design, small memory with high-speed SSD can replace large memory cache and achieve better performance.
  - Could directly use persistent memory to achieve the effect of this work?

Q&A