Centre for Research-based Innovation

Established by the Research Council of Norway

Efficient Query Processing in Distributed Search Engines

Simon Jonassen

Department of Computer and Information Science

Norwegian University of Science and Technology

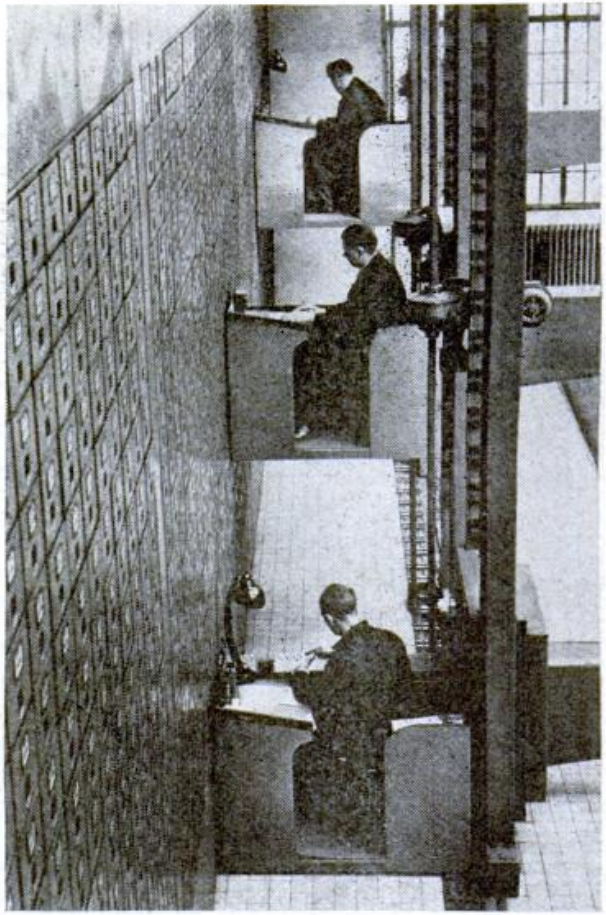
IAD INFORMATION ACCESS DISRUPTIONS

NTNU

Outline

- **Introduction**
  - Motivation
  - Research questions
  - Research process
- Part I – Partitioned query processing
- Part II – Skipping and pruning
- Part III – Caching
- Conclusions

Motivation



Clerks examining correspondence in the mammoth file of an insurance company

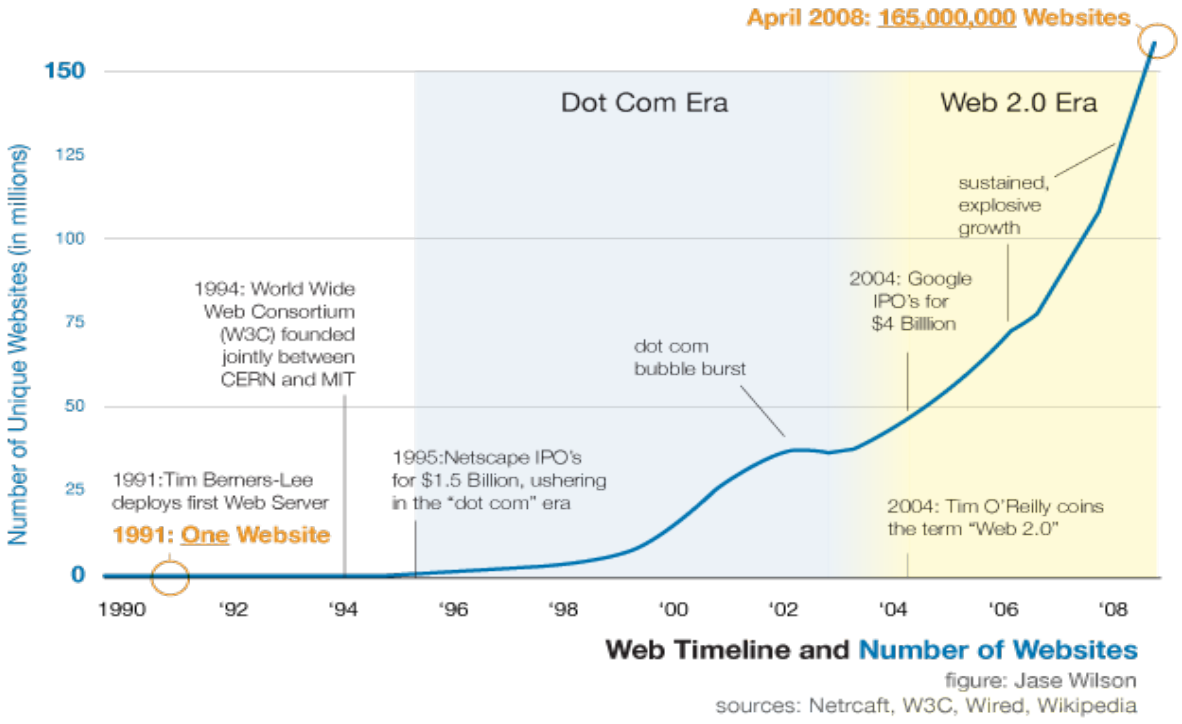
World's Largest File Has Traveling Desks

CLERKS ride up and down on "elevator desks" to consult the gigantic correspondence file of a Czechoslovakian insurance organization. Said to be the largest in the world, the huge letter file consists of 3,000 drawers, each ten feet long, covering 4,000 square feet of wall space and extending sixteen feet up from the floor. Clerks press control levers to move their desks up, down, or sidewise until they reach the desired file drawer, which opens automatically.

Popular Science, August 1937.

Motivation

- Web search: rapid information growth and tight performance constraints.
- Search is used in most of Web applications.



Web Timeline and Number of Websites

figure: Jase Wilson

sources: Netcraft, W3C, Wired, Wikipedia

Example: Google

- Estimated Web index size in 2012: 35-50 billion pages.
- More than a billion visitors in 2012.
- 36 data centers in 2008.
- Infrastructure cost in 4Q 2011: ~ 951 million USD.

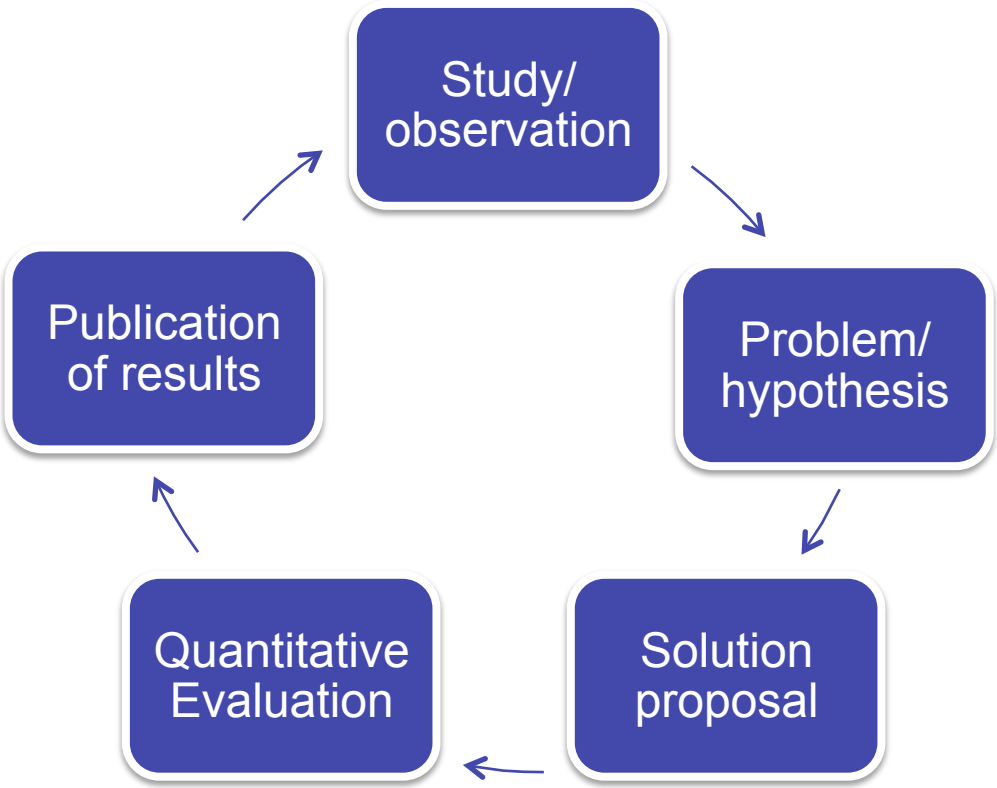
Research questions

- *What are the main challenges of existing methods for efficient query processing in distributed search engines and how can these be resolved?*

- **How to improve the efficiency of**
  - partitioned query processing
  - query processing and pruning
  - caching

Research process

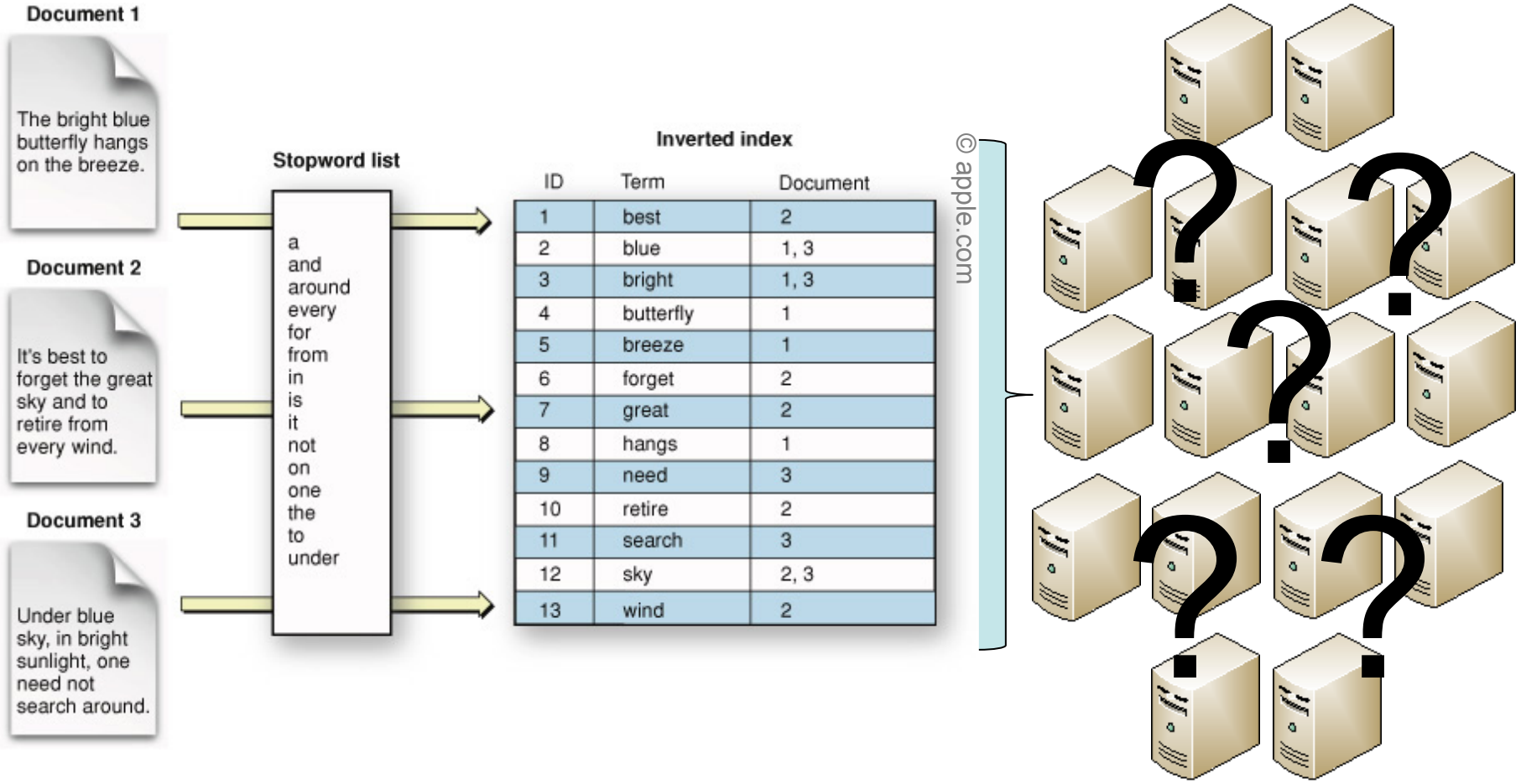
- Iterative, exploratory research:



Outline

- Introduction
- **Part I – Partitioned query processing**
  - Introduction
  - Motivation
  - Combined semi-pipelined query processing
  - Efficient compressed skipping for disjunctive text-queries
  - Pipelined query processing with skipping
  - Intra-query concurrent pipelined processing
  - Further work
- Part II – Skipping and pruning
- Part III – Caching
- Conclusions

Inverted index approach to IR

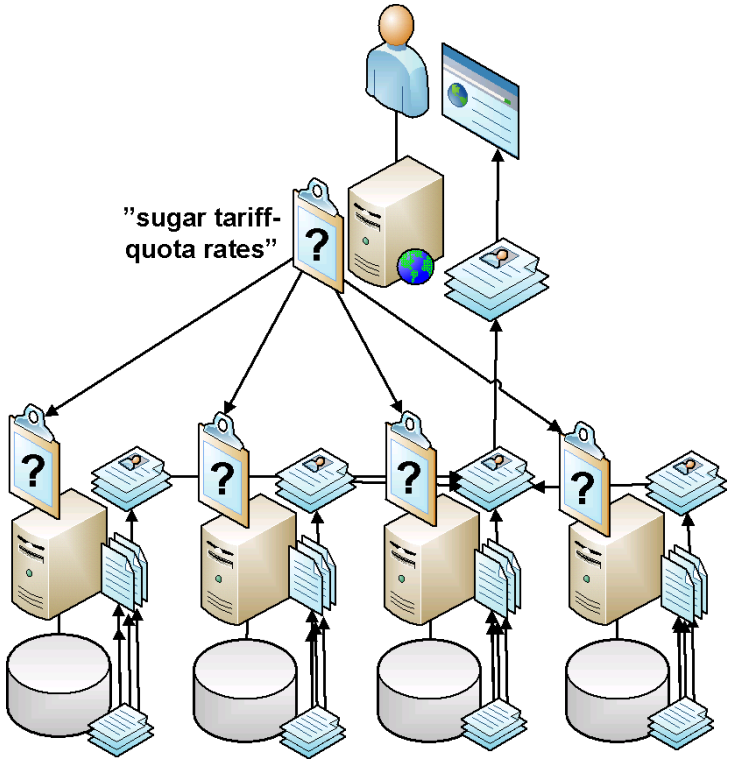


Document-wise partitioning

- Each query is processed by all of the nodes in parallel.
- One of the nodes merges the results.

- Advantages:
  - Simple, fast and scalable.
  - Intra-query concurrency.

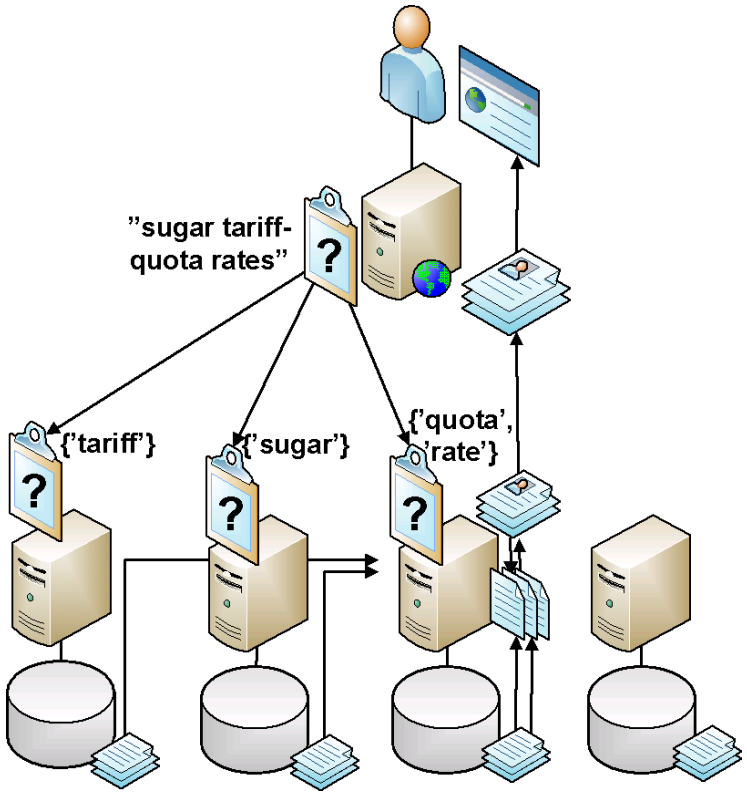
- Challenges:
  - Each query is processed by all of the nodes.
  - Number of disk seeks.



Term-wise partitioning

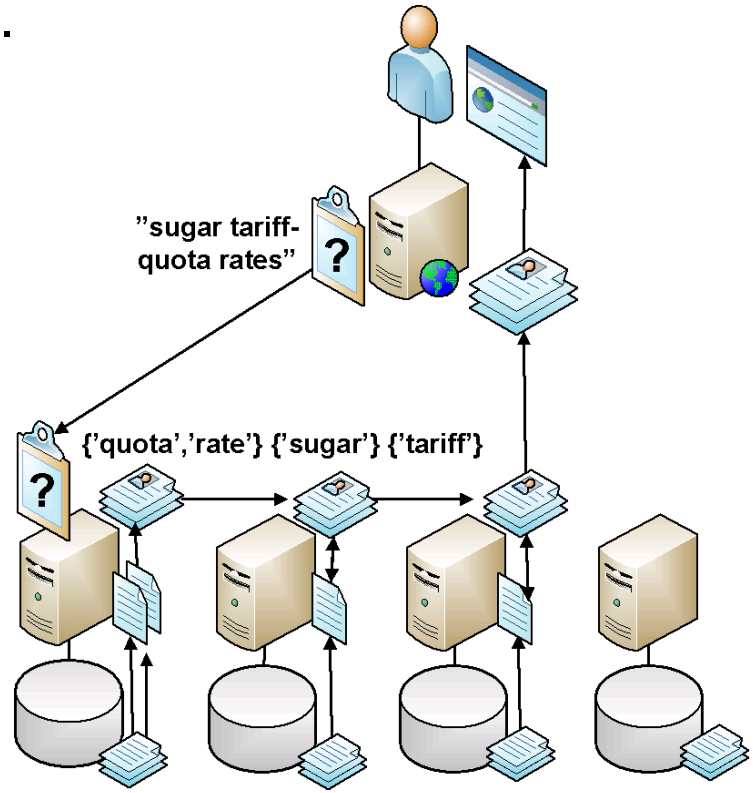
- Each node fetches posting lists and sends them to one of the nodes, which receives and processes all of them.

- Advantages:
  - A query involves only a few nodes.
  - Smaller number of disk seeks.
  - Inter-query concurrency.
- Challenges:
  - A single node does all processing.
  - Other nodes act just like advanced network disks.
  - High network load.
  - Load balancing.



Pipelined query processing [Moffat *et al.* 2007]

- A *query bundle* is routed from one node to next.
- Each node contributes with its own postings.
  The last node extracts the top results.
- The maximum accumulator set size
  is constrained. [Lester *et al.* 2005]



- Advantages:
  - The work is divided between the nodes.
  - Reduced overhead on the last node.
  - Reduced network load.
- Challenges:
  - Long query latency.
  - Load balancing.

Pipelined query processing (TP/PL)

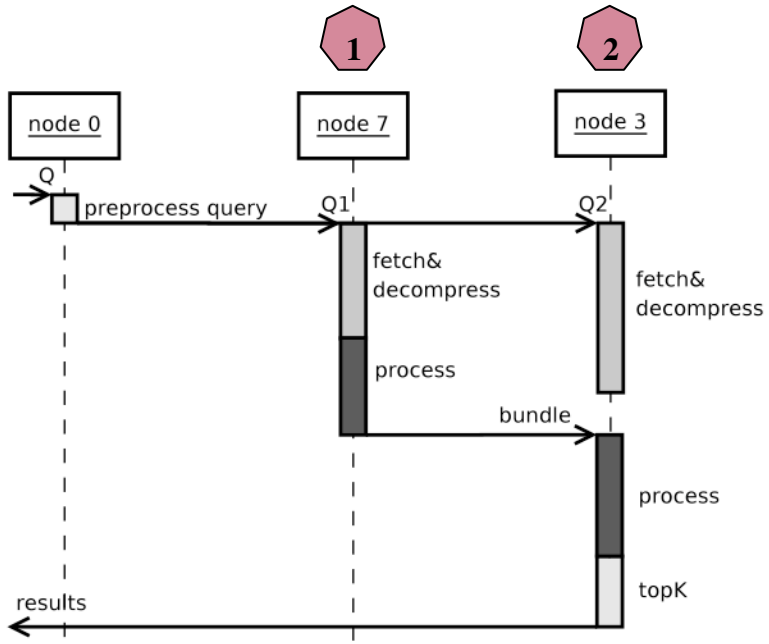
- Several publications find TP/PL being advantageous under certain conditions.
  - *Expensive disk access, fast network, efficient pruning, high multiprogramming level (query load), small main memory, etc.*
- TP/PL has several advantages.
  - *Number of involved nodes, inter-query concurrency, etc.*
- TP/PL may outperform DP in peak throughput.
  - *Requires high multiprogramming level and results in long query latency.*
- Practical results are more pessimistic.
  - ➢ ***How can we improve the method?***
- The main challenges of TP/PL [Büttcher *et al.* 2010]:
  - *Scalability, load balancing, term/node-at-a-time processing and lack of intra-query concurrency.*

Motivation

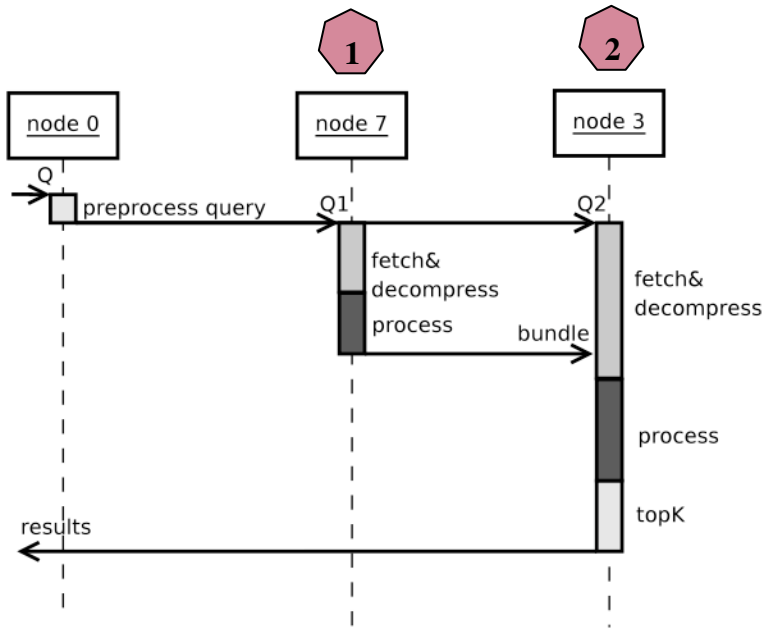
- Pipelined – higher throughput, but longer latency.
- Non-pipelined – shorter latency, but lower throughput.
- We wanted to combine the advantage of both methods: short latency AND high throughput.

Contributions

✓ Semi-pipelined query processing:



(a) Disk-Access/Decompression Latency Hiding



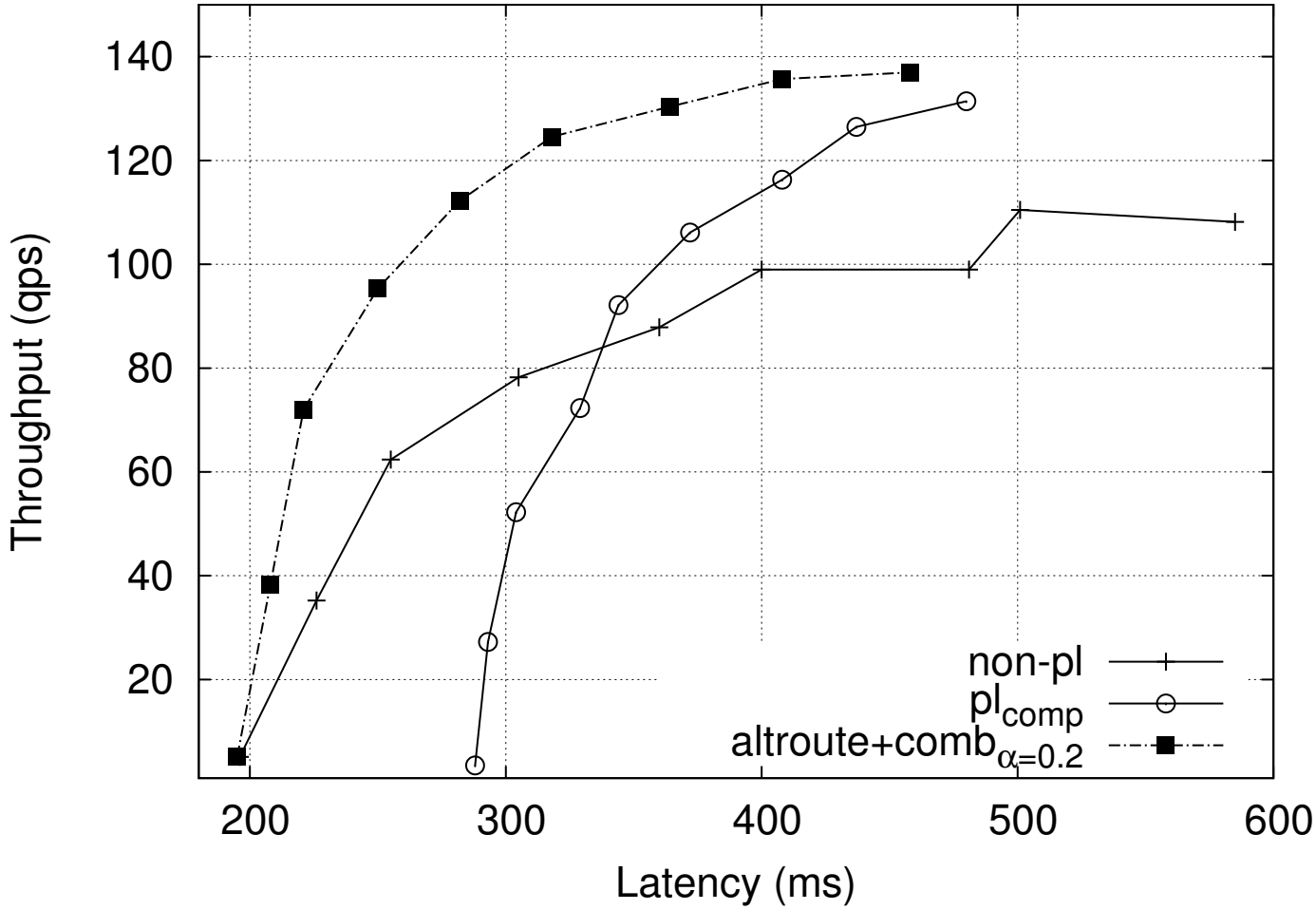
(b) Processing Latency Hiding

Contributions and evaluation

- ✓ Decision heuristic and alternative routing strategy.

- ✓ Evaluated using a modified version of Terrier v2.2 (http://terrier.org):
  - ➢ 426GB TREC GOV2 Corpus – 25 million documents.
  - ➢ 20 000 queries from the TREC Terabyte Track Efficiency Topics 2005 (first 10 000 queries were used as a warm-up).
  - ➢ 8 nodes with 2x2.0GHz Intel Quad-Core, 8GB RAM and a SATA hard-disk on each node. Gigabit network.

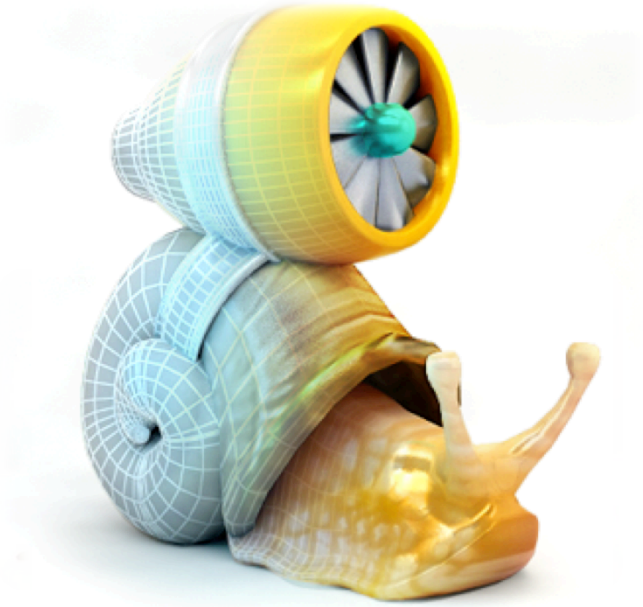
Main results



Post-note and motivation for other papers

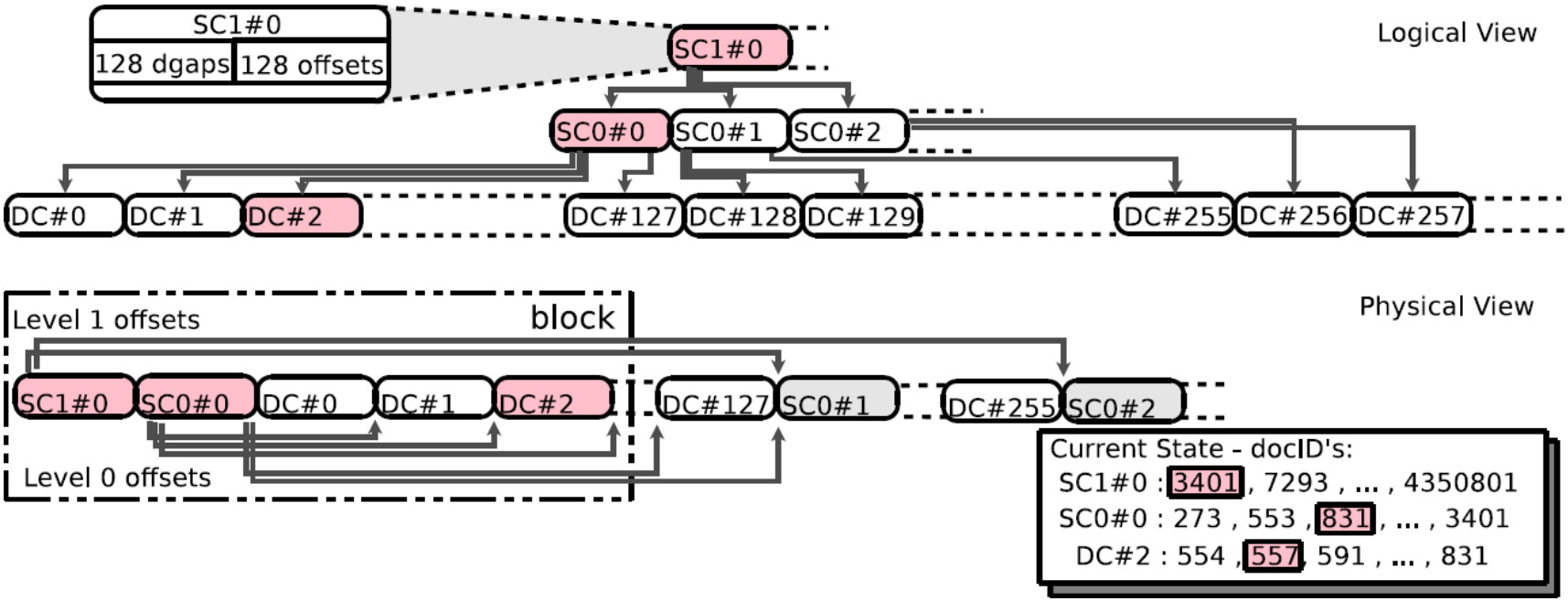
- "Inefficient" compression methods:
  - Gamma for doc ID gaps and Unary for frequencies.
- Posting lists have to be fetched and decompressed completely and remain in main memory until the sub-query has been fully processed.

- **Can we do this better?**
  - Modern "super-fast" index compression.
    - NewPFoR [Yan, Ding and Suel 2009].
  - Skipping.
  - Partial evaluation (pruning).



Contributions

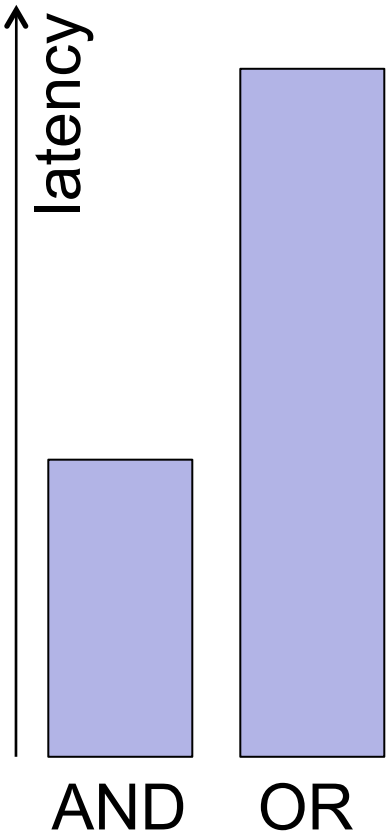
- ✓ A self-skipping inverted index designed specifically for NewPFoR compression.



Contributions

✓ Query processing methods for disjunctive (OR) queries:

- A skipping version of the space-limited pruning method by Lester *et al.* 2007.
- A description of Max-Score that combines the advantage of the descriptions by Turtle and Flood 1995 and Strohman, Turtle and Croft 2005.
  - We also explained how to apply skipping and presented updated performance results.

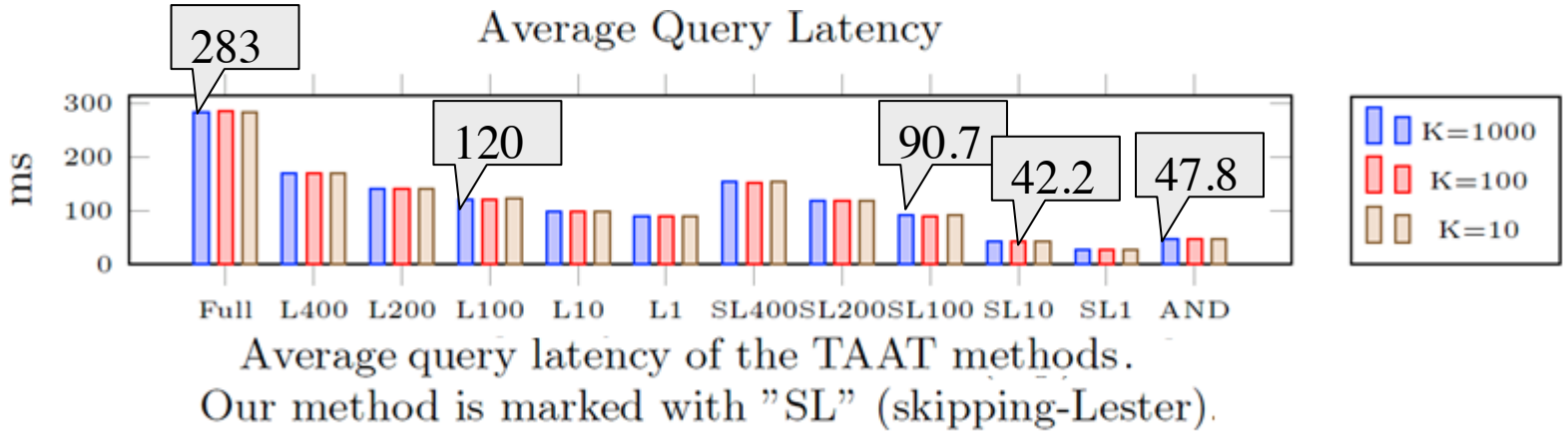


Evaluation

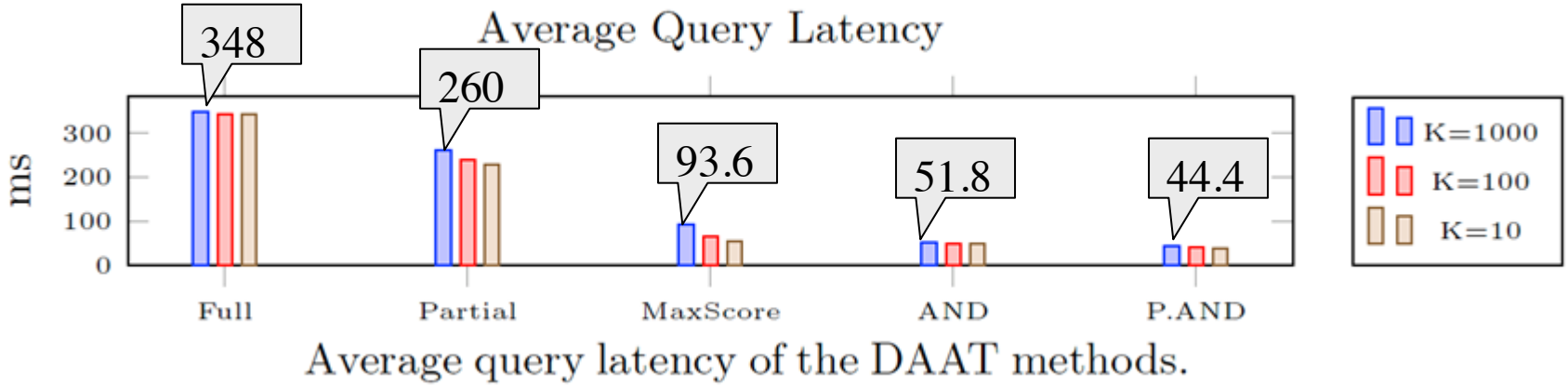
- ✓ Evaluated with a real implementation:
  - ➢ TREC GOV2.
    - 15.4 million terms, 25.2 million docs and 4.7 billion postings.
    - Index w/o skipping is 5.977GB. Skipping adds 87.1MB.
  - ➢ TREC Terabyte Track Efficiency Topics 2005.
    - 10 000 queries with 2+ query terms.
  - ➢ Workstation: 2.66GHz Intel Quad-Core 2, 8GB RAM, 1TB 7200RPM SATA2. 16KB blocks by default.

Part I: Efficient compressed skipping for disjunctive text-queries.

Main results



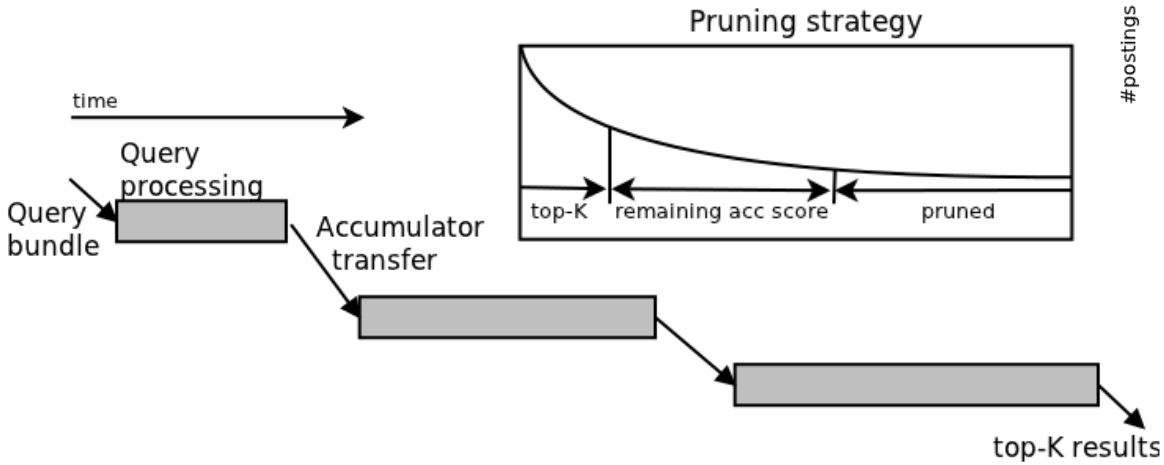
Average query latency of the TAAT methods.
Our method is marked with "SL" (skipping-Lester).

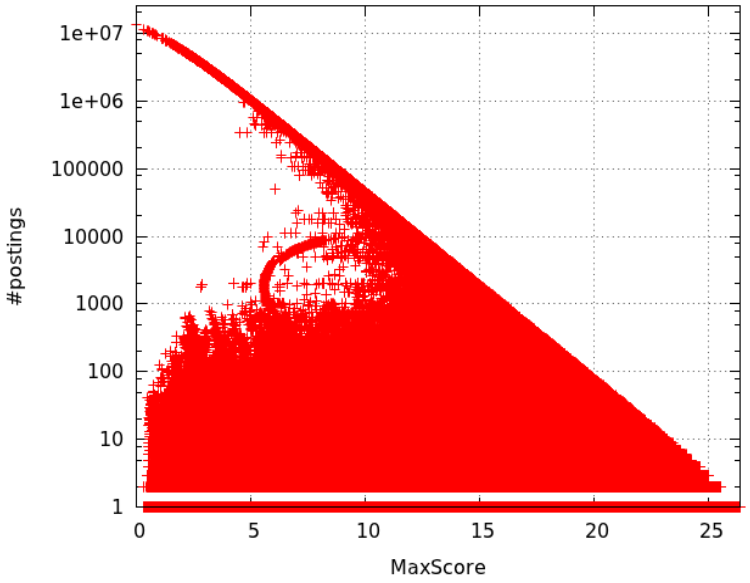


Average query latency of the DAAT methods.

Contributions

- ✓ Pipelined query processing with skipping.
- ✓ Pipelined Max-Score with DAAT sub-query processing.
- ✓ Posting list assignment by maximum scores.

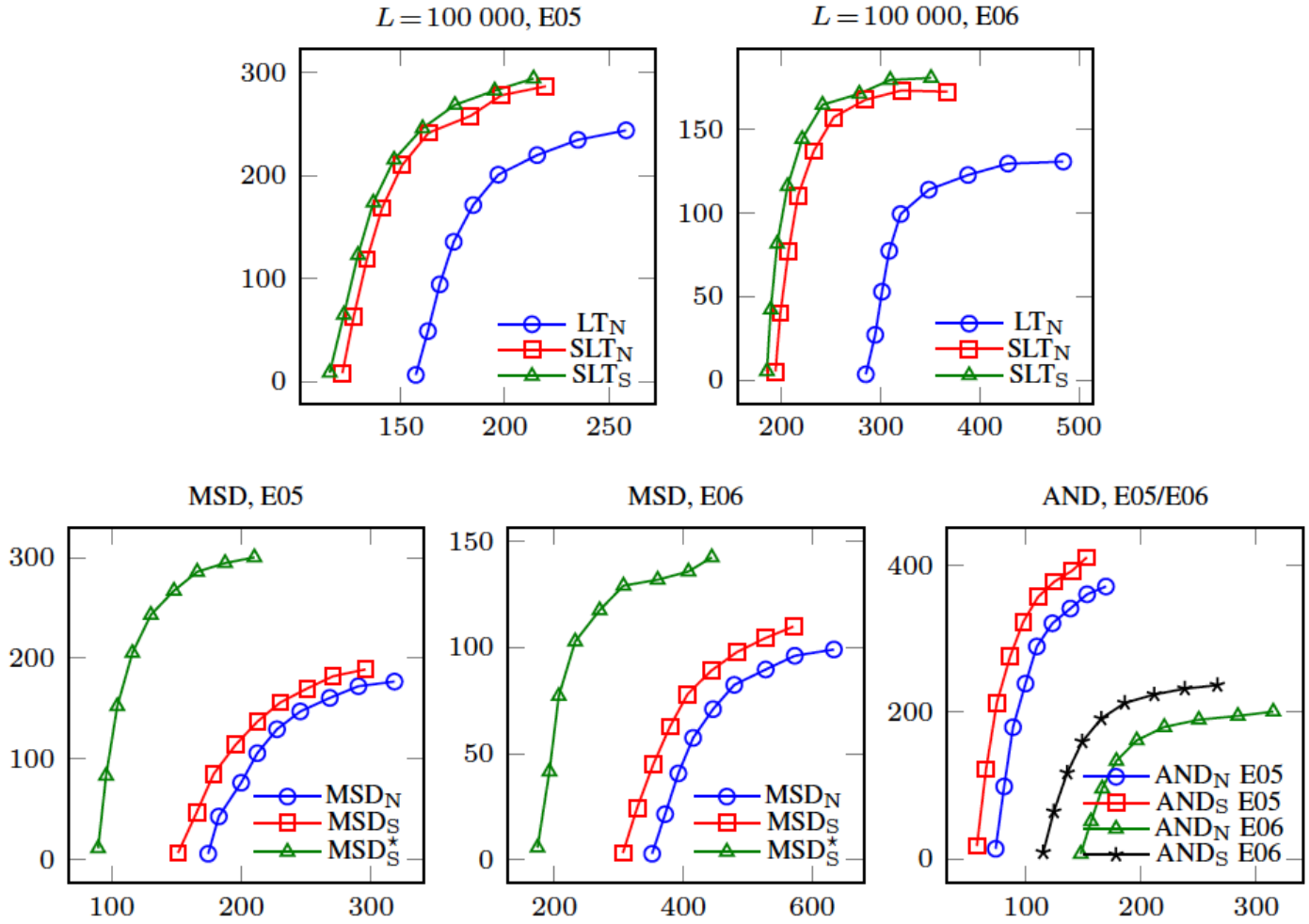




Evaluation

- Evaluated with a real implementation:
  - Same index design as in the last paper, but now extended to a distributed system.
  - TREC GOV2 and Terabyte Track Efficiency Topics 2005/2006:
    - 5 000 queries – warm-up, 15 000 – evaluation.
  - 9 node cluster – same as in the first paper, plus a broker.

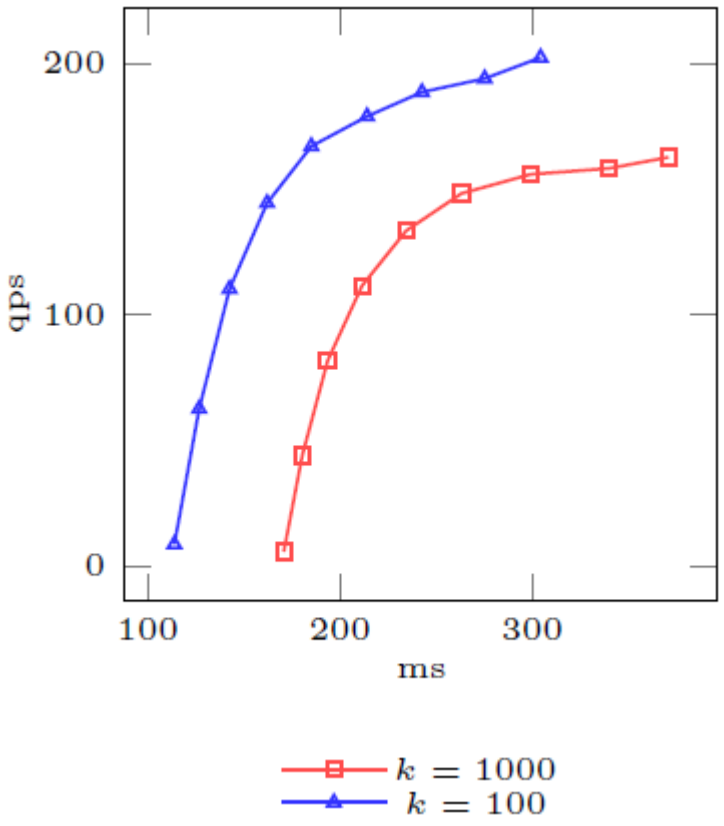
Main results



Throughput (y-axis, qps) and latency (ms) with varied multiprogramming.

Motivation

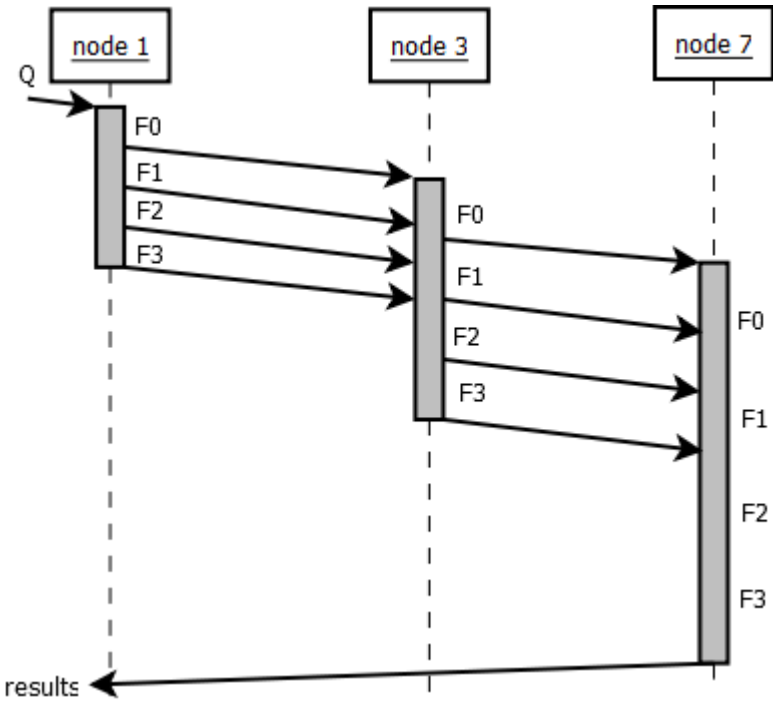
- At low query loads there is available CPU/network capacity.
- In this case, the performance can be improved by intra-query concurrency.
- **How can we provide intra-query concurrency with TP/PL?**



Contributions

✓ Parallel sub-query execution on different nodes.

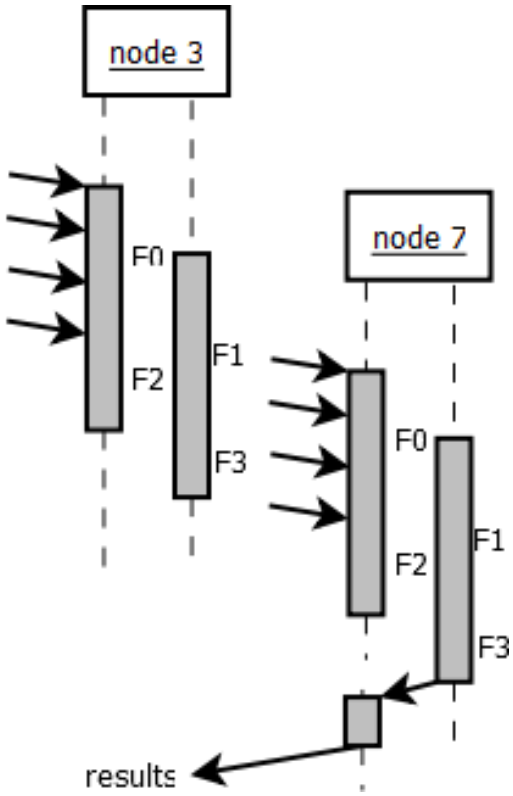
- Fragment-based processing:
- Each fragment covers a document ID range.
- The number of fragments in each query depends on the estimated processing cost.



Contributions

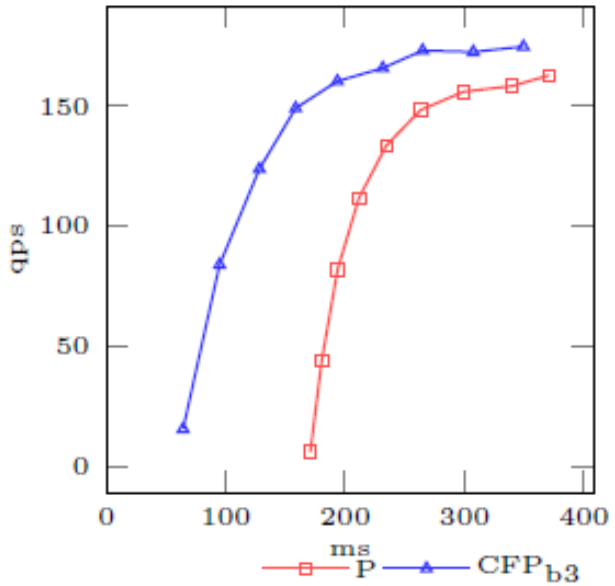
✓ Concurrent sub-query execution on the same node.

- When the query load is low, different fragments of the same query can be processed concurrently.
- Concurrency depends on the number of fragments in the query and the load on the node.

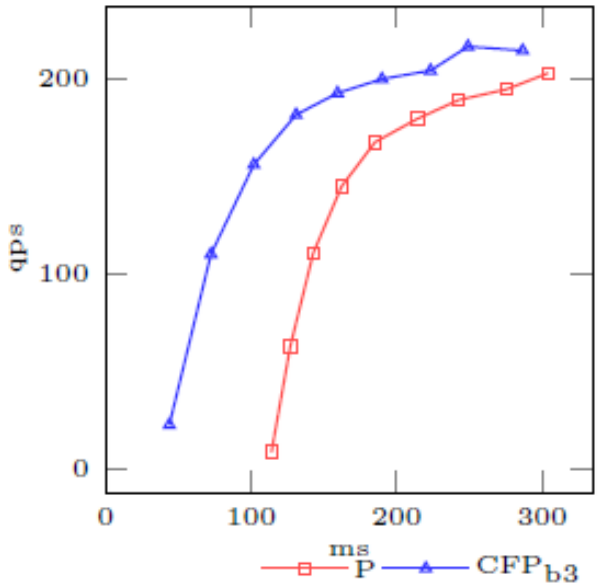


Evaluation and main results

- ✓ Evaluated with TREC GOV2, TREC Terabyte Track Efficiency Topics 2005 (2 000 warm-up and 8 000 test queries).



CFP$_{b3}$/P, $k = 1000$



CFP$_{b3}$/P, $k = 100$

Throughput and latency with the concurrent fragment pipeline (CFP).

Part I.

Further work

- Evaluation of scalability:
  - So far: 8 nodes and 25 mil documents.
- Further extension of the methods:
  - Dynamic load balancing.
  - Caching.
- Open questions:
  - Term- vs document-wise partitioning.
  - Document- vs impact-ordered indexes.

Outline

- Introduction
- Part I – Partitioned query processing
- **Part II – Skipping and pruning**
  - Contributions presented within Part I
  - Improving pruning efficiency via linear programming
- Part III – Caching
- Conclusions

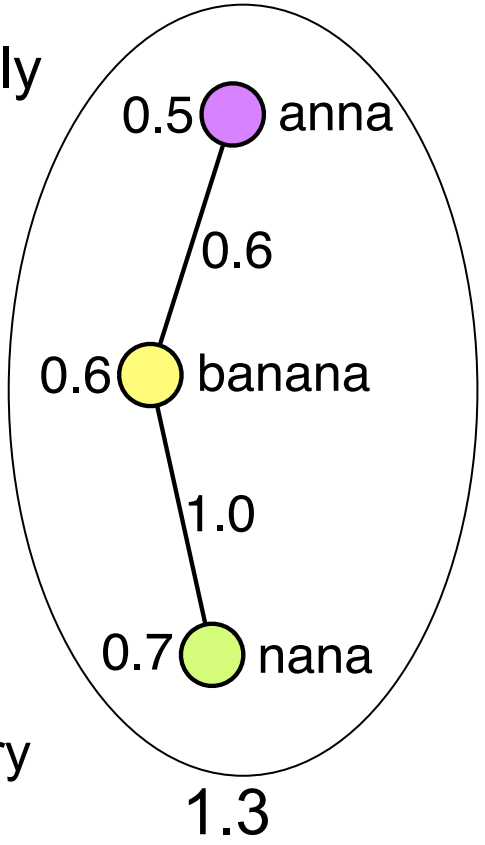
Part II.

Contributions presented within Part I

- ✓ Efficient processing of disjunctive (OR) queries (monolithic index).
- ✓ Pipelined query processing with skipping (term-wise partitioned index).

Motivation and contributions

- Pruning methods s.a. Max-Score and WAND rely on tight score upper-bound estimates.
- The upper-bound for a combination of several terms is smaller than the sum of the individual upper-bounds.
- Terms and their combinations repeat.



✓ LP approach to improving the performance of Max-Score/WAND.

- ➢ Relies on LP estimation of score bounds for query subsets.
- ➢ In particular, uses upper bounds of single terms and the most frequent term-pairs.

Part II: Improving dynamic index pruning via linear programming.

Evaluation and extended results

- ✓ Evaluated with a real implementation.
  - ➢ 20 million Web pages, 100 000 queries from Yahoo, Max-Score.
- ✓ The results indicate significant I/O, decompression and computational savings for queries of 3 to 6 terms.

Latency improvements.

	|q|	T (ms)	TI	TI**
AND	**3**	10.759	1.59%	1.64%
(k=10)	**4**	10.589	1.30%	1.43%
	5	9.085	0.09%	0.97%
	6	6.256	-14.86%	-0.95%
OR	**3**	18.448	3.74%	3.78%
(k=10)	**4**	24.590	6.85%	6.93%
	5	33.148	7.85%	8.24%
	6	42.766	6.04%	9.08%

Query length distribution.

|q|	%
1	14.91%
2	36.80%
3	29.75%
4	12.35%
5	4.08%
6	1.30%
7	0.42%
8+	0.39%

LP solver latency.

|q|	AND (ms)	OR (ms)
3	0.002	0.002
4	0.01	0.016
5	0.082	0.136
6	0.928	1.483
7	14.794	22.676
8	300.728	437.005

** 10x faster LP solver.

Further work

- Larger document collection.
- Other types of queries.
- Reuse of previously computed scores.
- Application to pipelined query processing.

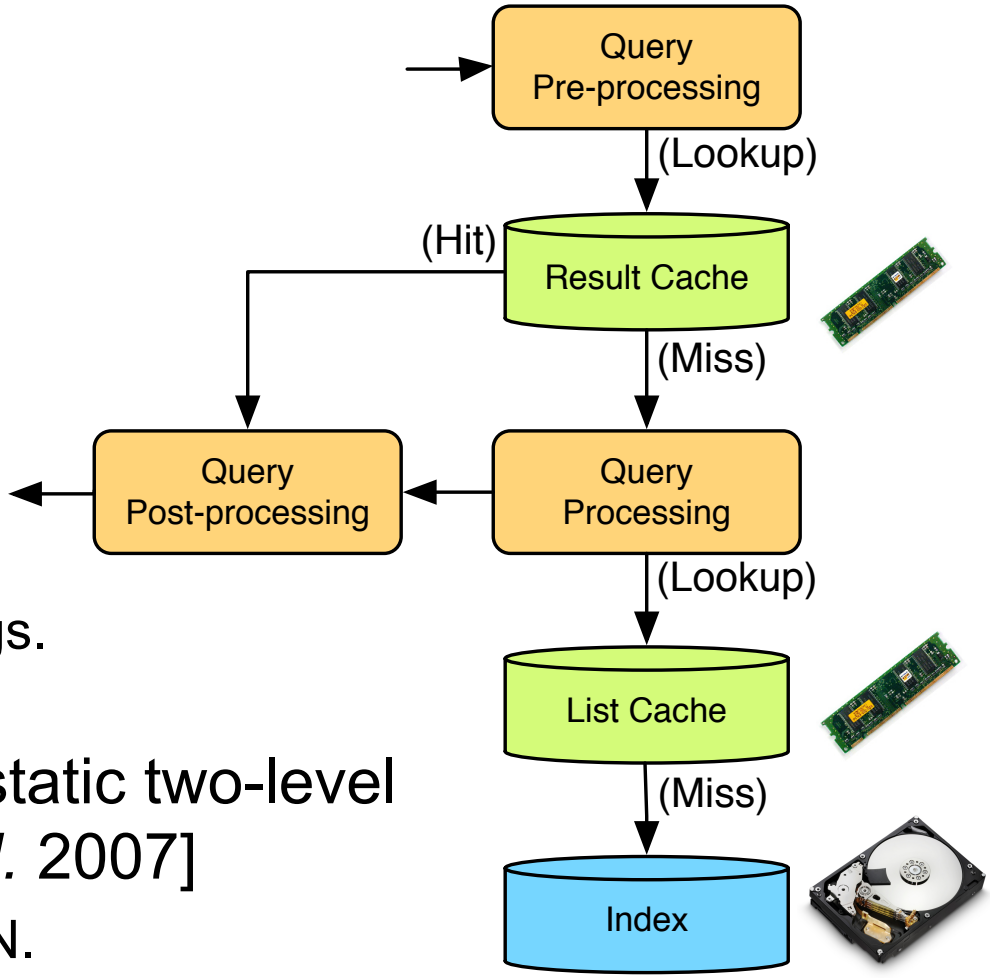
Outline

- Introduction
- Part I – Partitioned query processing
- Part II – Skipping and pruning
- **Part III – Caching**
  - Modeling static caching in Web search engines
  - Prefetching query results and its impact on search engines
- Conclusions

Motivation

- Result cache:
  - Lower hit rate.
  - Less space consuming.
  - Significant time savings.
- List cache:
  - Higher hit rate.
  - More space consuming.
  - Less significant time savings.

- Finding optimal split in a static two-level cache. [Baeza-Yates *et al.* 2007]
  - Scenarios: local, LAN, WAN.



Part III: Modeling static caching in Web search engines.

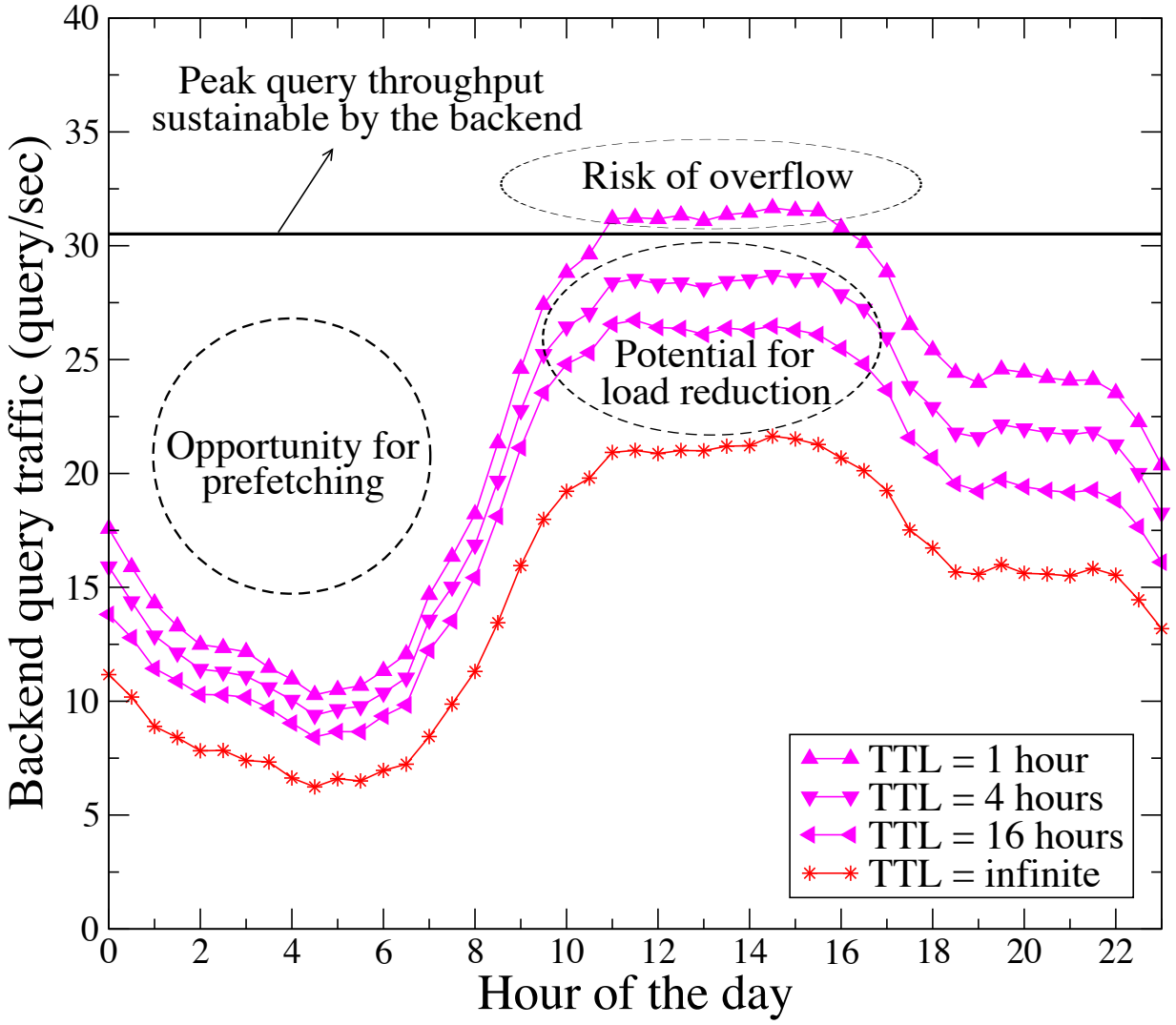
Contributions, evaluation and further work

- ✓ A modeling approach that allows to find a nearly optimal split in a static two-level cache.
  - ➢ Requires estimation of only six parameters.
- ✓ Evaluated with a cache-simulator:
  - ➢ 37.9 million documents from UK domain (2006)
  - ➢ 80.7 million queries (April 2006 - April 2007)
- Further work:
  - Larger query log and document collection.
  - Improved model.
  - Other types of cache.

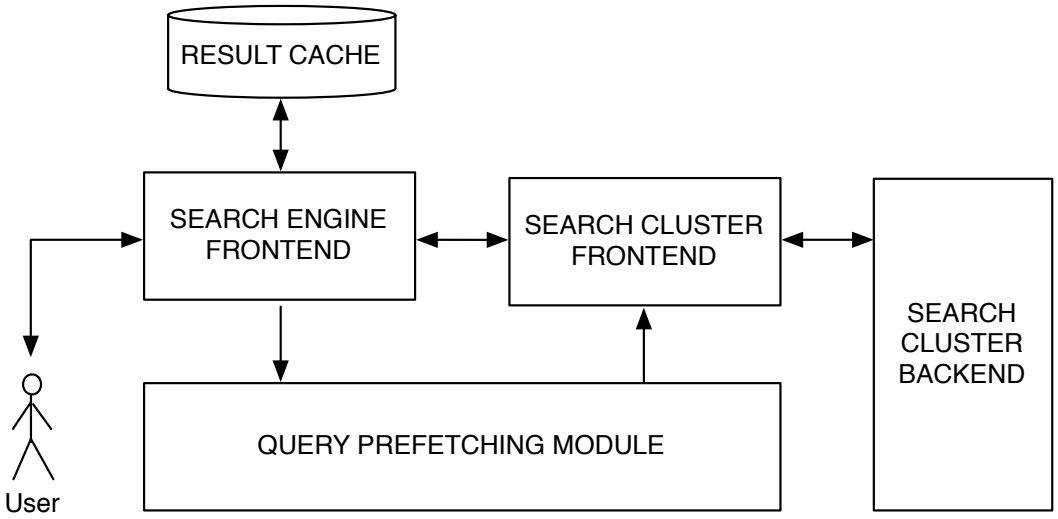
Background and related work

- Result caching:
  - Reuse of previously computed results to answer future queries.
    - Our data: About 60% of queries repeat.
  - Limited vs infinite-capacity caches.
    - Limited: Beaza-Yates *et al.* 2007, Gan and Suel 2009.
    - Infinite: Cambazoglu *et al.* 2010.
  - Cache staleness problem.
    - Results become *stale* after a while.
    - Time-to-live (TTL) invalidation.
      - Cambazoglu *et al.* 2010, Alici *et al.* 2012.
    - Result freshness.
      - Cambazoglu *et al.* 2010.

Motivation


Peak query throughput
sustainable by the backend
Risk of overflow
Potential for
load reduction
Opportunity for
prefetching
Backend query traffic (query/sec)
Hour of the day
TTL = 1 hour
TTL = 4 hours
TTL = 16 hours
TTL = infinite
0
5
10
15
20
25
30
35
40
0
2
4
6
8
10
12
14
16
18
20
22

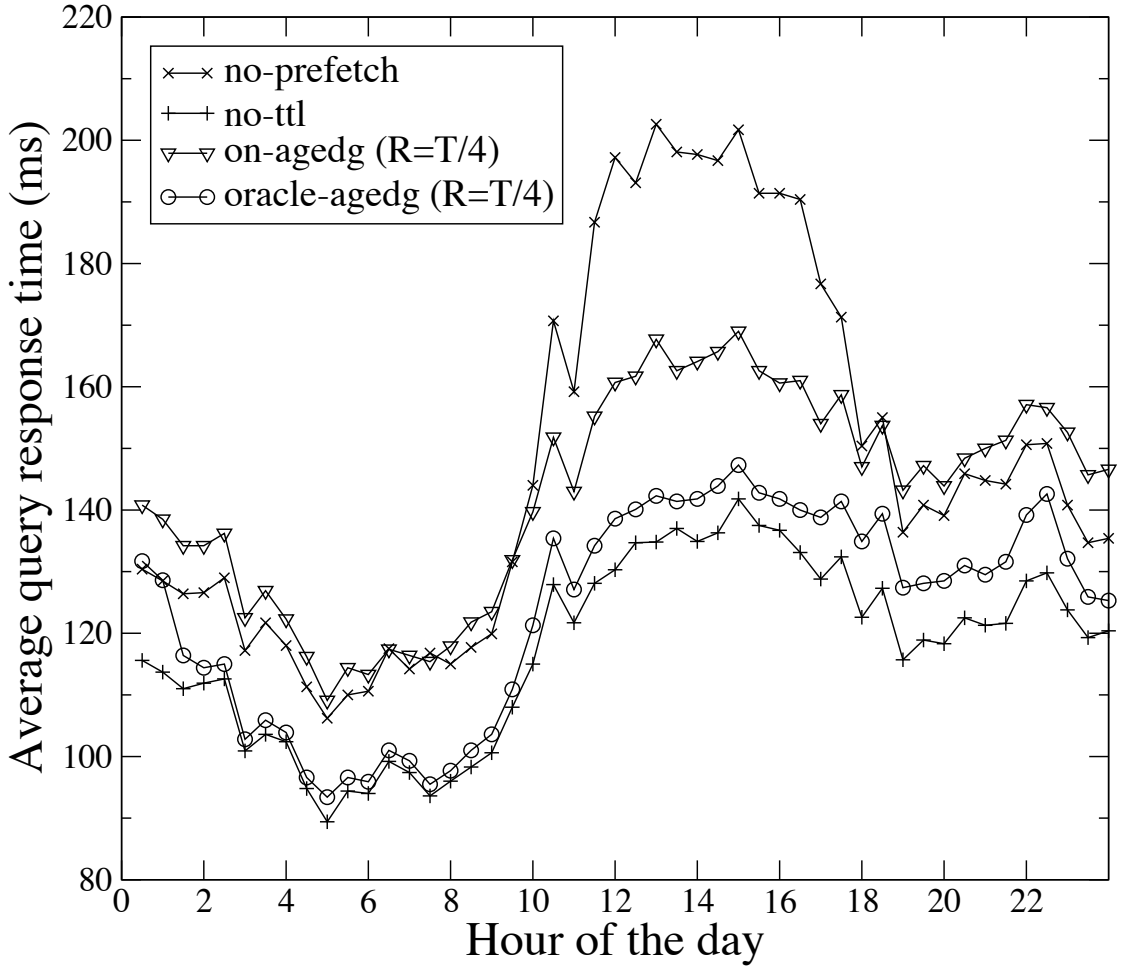
Contributions



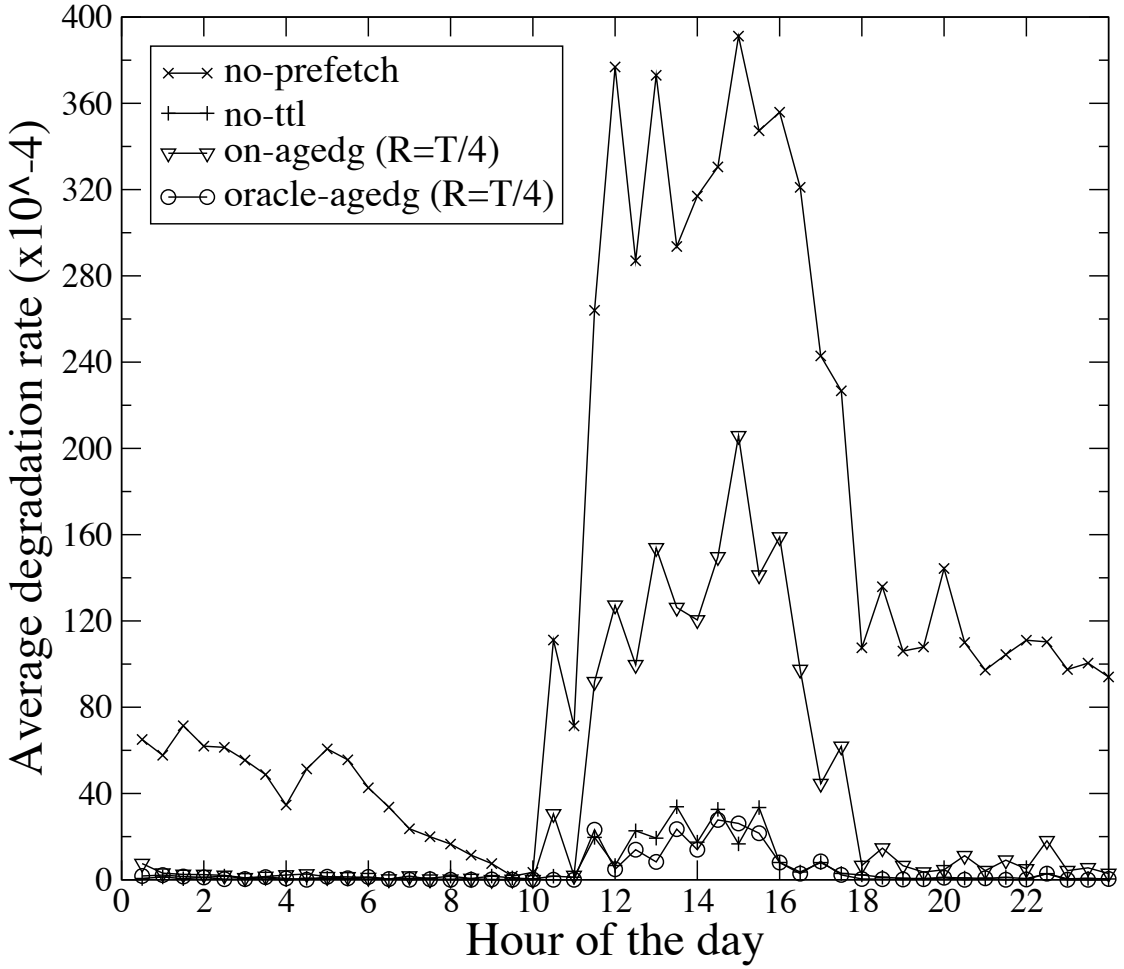
- ✓ Three-stage approach:
  1. Selection – *Which query results to prefetch?*
  2. Ordering – *Which query results to prefetch first?*
  3. Scheduling – *When to prefetch?*
- ✓ Two selection and ordering strategies:
  - *Offline* – queries issued 24 hours before.
  - *Online* – machine learned prediction of the next arrival.

Evaluation and main results

- ✓ Evaluated with an event-driven search engine simulator and Web queries sampled Yahoo! Logs.
  - ➢ Millions of queries, billions of documents, thousands of nodes.
  - ➢ Methods: no-ttl, no-prefetch, age-dg, online, offline.
- ✓ The results show that prefetching can improve the hit rate, query response time, result freshness and query degradation rate relative to the state-of-the-art baseline.
- ✓ The key aspect is the prediction methodology.



Average query response time (1020 nodes).



Average degradation (850 nodes).

Further work

- Improved prediction model.
- Speculative prefetching.
- Economic evaluation.

Outline

- Introduction
- Part I – Partitioned query processing
- Part II – Skipping and pruning
- Part III – Caching
- **Conclusions**

Conclusions

- We have contributed to resolving some of the most important challenges in distributed query processing.
- Our results indicate significant improvements over the state-of-the-art methods.
- Practical applicability and implications, however, need a further evaluation in real-life settings.
- There are several opportunities for future work.

Key references

- Alici at al.: “Adaptive Time-to-Live Strategies for Query Result Caching in Web Search Engines”. *In Proc.* ECIR 2012, pp.401-412.
- Baeza-Yates et al.: “The impact of caching on search engines”. *In Proc.* SIGIR 2007, pp.183-190.
- Büttcher, Clarke and Cormack: “Information Retrieval – Implementing and Evaluating Search Engines”, 2010.
- Cambazoglu et al.: “A refreshing perspective of search engine caching”. *In Proc. WWW* 2010, pp. 181-190.
- Gan and Suel: “Improved techniques for result caching in web search engines”. *In Proc.* WWW 2009, pp. 431-440.
- Lester et al.: “Space-Limited Ranked Query Evaluation Using Adaptive Pruning”. *In Proc.* WISE 2005, pp. 470-477.
- Moffat et al.: “A pipelined architecture for distributed text query evaluation”. *Inf. Retr.* 10(3) 2007, pp. 205-231.
- Strohman, Turtle and Croft: “Optimization strategies for complex queries”. *In Proc.* SIGIR 2005, pp. 175-182.
- Turtle and Flood: “Query Evaluation: Strategies and Optimizations*”. Inf. Process. Manage.* 31(6) 1995, pp. 838-849.
- Yan, Ding and Suel: “Inverted index compression and query processing with optimized document ordering”. *In Proc.* WWW 2009, pp. 401-410.

Publications

1. Baeza-Yates and Jonassen: “Modeling Static Caching in Web Search Engines”. *In Proc.* ECIR 2012, pp. 436-446.
2. Jonassen and Bratsberg: “Improving the Performance of Pipelined Query Processing with Skipping”. *In Proc.* WISE 2012, pp. 1-15.
3. Jonassen and Bratsberg: “Intra-Query Concurrent Pipelined Processing for Distributed Full-Text Retrieval”. *In Proc.* ECIR 2012, pp. 413-425.
4. Jonassen and Bratsberg: “Efficient Compressed Inverted Index Skipping for Disjunctive Text-Queries”. *In Proc.* ECIR 2011, pp. 530-542.
5. Jonassen and Bratsberg: “A Combined Semi-Pipelined Query Processing Architecture for Distributed Full-Text Retrieval”. *In Proc. WISE* 2010, pp. 587-601.
6. Jonassen and Cambazoglu: “Improving Dynamic Index Pruning via Linear Programming”. *In progress.*
7. Jonassen, Cambazoglu and Silvestri: “Prefetching Query Results and its Impact on Search Engines”. *In Proc.* SIGIR 2012, pp. 631-640.

Secondary (Not included):

1. Cambazoglu et al.: “A Term-Based Inverted Index Partitioning Model for Efficient Query Processing”. *Under submission.*
2. Rocha-Junior et al.: “Efficient Processing of Top-k Spatial Keyword Queries”. *In Proc.* SSTD 2011, pp. 205-222.
3. Jonassen: “Scalable Search Platform: Improving Pipelined Query Processing for Distributed Full-Text Retrieval”. *In Proc.* WWW (Comp. Vol.) 2012, pp. 145-150.
4. Jonassen and Bratsberg: “Impact of the Query Model and System Settings on Performance of Distributed Inverted Indexes”. *In Proc.* NIK 2009, pp. 143-151.

Thanks!!!

IAD
INFORMATION ACCESS DISRUPTIONS

NTNU