max planck institut
informatik

Top-k Aggregation Queries in Large-Scale Distributed Systems

Sebastian Michel

Before: Philipps Universität Marburg (Diplom)
Depuis août 2007: Ecole Polytechnique Fédérale de Lausanne (Postdoc)

sebastian.michel@epfl.ch

2009/03/06

Example: One Access Log for the Whole Network

Server	ClientIP	Bytes
www.server1.com	192.168.1.3	17kB
www.server2.com	192.168.1.1	9kB
www.server1.com	192.168.1.4	12kB
www.server1.com	192.168.1.2	11kB
www.server3.com	192.168.1.3	12kB
www.server1.com	192.168.1.5	4kB
www.server2.com	192.168.1.3	7kB
www.server3.com	192.168.1.5	5kB
www.server2.com	192.168.1.2	2kB
www.server2.com	192.168.1.6	1kB
www.server2.com	192.168.1.7	1kB
www.server3.com	192.168.1.1	19kB
www.server3.com	192.168.1.4	15kB
www.server1.com	192.168.1.6	2kB
www.server3.com	192.168.1.7	2kB

Top-k Query Processing

```
SELECT ClientIP, Aggr(Bytes)
FROM AccessLog
GROUP BY ClientIP
ORDER BY Aggr(Bytes)
LIMIT k
```

Goal

Return the top-k items that have the highest aggregated number of bytes.

Example: Access Logs at Different Servers

www.server1.com:

ClientIP	Bytes
192.168.1.3	17kB
192.168.1.4	12kB
192.168.1.2	11kB
192.168.1.5	4kB
192.168.1.6	2kB

www.server2.com:

ClientIP	Bytes
192.168.1.1	9kB
192.168.1.3	7kB
192.168.1.2	2kB
192.168.1.6	1kB
192.168.1.7	1kB

www.server3.com:

ClientIP	Bytes
192.168.1.1	19kB
192.168.1.4	15kB
192.168.1.3	12kB
192.168.1.5	5kB
192.168.1.7	2kB

Other Application Scenarios

- Peer-to-Peer web search (distributed term indices)
- Sensor networks

Computational Model & Data Aggregation

Computational Model

- Query $Q = \{t_1, t_2, ..., t_m\}$ with m attributes.
- Index lists $l_1, l_2, ..., l_m$ spread across m Peers $P_1, ..., P_m$.
- $(itemId, score)$-pairs sorted by "score" descending.

Score Aggregation

- Monotonic aggregation function (usually summation).

Computing Score Bounds

Partial score

$$worstscore(d) = agg(seenvalues)$$

Consider the item that is currently at rank k, with score $min\text{-}k$.

Upper bound score

Best possible score

$$bestscore = worstscore + \sum_{i \notin E(d)} l_i(scandepth_i)$$

Pruning strategy

Dismiss items d with

$$bestscore(d) < min\text{-}k$$

Top-k Query Execution Algorithms ...

Well understood in centralized settings:

- Most prominent: Family of threshold algorithms
- Fagin's algorithms (FA, TA, NRA, CA)
- Extensions with probabilistic pruning
- ....

Query Processing Example (NRA)

www.server1.com:

ClientIP	Bytes
192.168.1.3	17kB
192.168.1.4	12kB
192.168.1.2	11kB
192.168.1.5	4kB
192.168.1.6	2kB

www.server2.com:

ClientIP	Bytes
192.168.1.1	9kB
192.168.1.3	7kB
192.168.1.2	2kB
192.168.1.6	1kB
192.168.1.7	1kB

www.server3.com:

ClientIP	Bytes
192.168.1.1	19kB
192.168.1.4	15kB
192.168.1.3	12kB
192.168.1.5	5kB
192.168.1.7	2kB

Candidates:

Id **worstscore** **bestscore**

min-k =?

Query Processing Example (NRA)

www.server1.com:

ClientIP	Bytes
192.168.1.3	17kB
192.168.1.4	12kB
192.168.1.2	11kB
192.168.1.5	4kB
192.168.1.6	2kB

www.server2.com:

ClientIP	Bytes
192.168.1.1	9kB
192.168.1.3	7kB
192.168.1.2	2kB
192.168.1.6	1kB
192.168.1.7	1kB

www.server3.com:

ClientIP	Bytes
192.168.1.1	19kB
192.168.1.4	15kB
192.168.1.3	12kB
192.168.1.5	5kB
192.168.1.7	2kB

Status of the Query Processing:

Candidates:

Id	worstscore	bestscore
192.168.1.3	17	-

$min\text{-}k = 17$

Query Processing Example (NRA)

www.server1.com:

ClientIP	Bytes
192.168.1.3	17kB
192.168.1.4	12kB
192.168.1.2	11kB
192.168.1.5	4kB
192.168.1.6	2kB

www.server2.com:

ClientIP	Bytes
192.168.1.1	9kB
192.168.1.3	7kB
192.168.1.2	2kB
192.168.1.6	1kB
192.168.1.7	1kB

www.server3.com:

ClientIP	Bytes
192.168.1.1	19kB
192.168.1.4	15kB
192.168.1.3	12kB
192.168.1.5	5kB
192.168.1.7	2kB

Status of the Query Processing:

Candidates:

Id	worstscore	bestscore
192.168.1.3	17	-
192.168.1.1	9	-

$min\text{-}k = 17$

Query Processing Example (NRA)

www.server1.com:

ClientIP	Bytes
192.168.1.3	17kB
192.168.1.4	12kB
192.168.1.2	11kB
192.168.1.5	4kB
192.168.1.6	2kB

www.server2.com:

ClientIP	Bytes
192.168.1.1	9kB
192.168.1.3	7kB
192.168.1.2	2kB
192.168.1.6	1kB
192.168.1.7	1kB

www.server3.com:

ClientIP	Bytes
192.168.1.1	19kB
192.168.1.4	15kB
192.168.1.3	12kB
192.168.1.5	5kB
192.168.1.7	2kB

Status of the Query Processing:

$min\text{-}k = 28$

Candidates:

Id	worstscore	bestscore
192.168.1.3	17	45
192.168.1.1	28	45

Query Processing Example (NRA)

www.server1.com:

ClientIP	Bytes
192.168.1.3	17kB
192.168.1.4	12kB
192.168.1.2	11kB
192.168.1.5	4kB
192.168.1.6	2kB

www.server2.com:

ClientIP	Bytes
192.168.1.1	9kB
192.168.1.3	7kB
192.168.1.2	2kB
192.168.1.6	1kB
192.168.1.7	1kB

www.server3.com:

ClientIP	Bytes
192.168.1.1	19kB
192.168.1.4	15kB
192.168.1.3	12kB
192.168.1.5	5kB
192.168.1.7	2kB

Status of the Query Processing:

$min\text{-}k = 28$

Candidates:

Id	worstscore	bestscore
192.168.1.3	17	45
192.168.1.1	28	40
192.168.1.4	12	40

Query Processing Example (NRA)

www.server1.com:

ClientIP	Bytes
192.168.1.3	17kB
192.168.1.4	12kB
192.168.1.2	11kB
192.168.1.5	4kB
192.168.1.6	2kB

www.server2.com:

ClientIP	Bytes
192.168.1.1	9kB
192.168.1.3	7kB
192.168.1.2	2kB
192.168.1.6	1kB
192.168.1.7	1kB

www.server3.com:

ClientIP	Bytes
192.168.1.1	19kB
192.168.1.4	15kB
192.168.1.3	12kB
192.168.1.5	5kB
192.168.1.7	2kB

Status of the Query Processing:

$min\text{-}k = 28$

Candidates:

Id	worstscore	bestscore
192.168.1.3	24	43
192.168.1.1	28	40
192.168.1.4	12	38

Query Processing Example (NRA)

www.server1.com:

ClientIP	Bytes
192.168.1.3	17kB
192.168.1.4	12kB
192.168.1.2	11kB
192.168.1.5	4kB
192.168.1.6	2kB

www.server2.com:

ClientIP	Bytes
192.168.1.1	9kB
192.168.1.3	7kB
192.168.1.2	2kB
192.168.1.6	1kB
192.168.1.7	1kB

www.server3.com:

ClientIP	Bytes
192.168.1.1	19kB
192.168.1.4	15kB
192.168.1.3	12kB
192.168.1.5	5kB
192.168.1.7	2kB

Status of the Query Processing:

$min\text{-}k = 28$

Candidates:

Id	worstscore	bestscore
192.168.1.3	24	39
192.168.1.1	28	40
192.168.1.4	27	34

Query Processing Example (NRA)

www.server1.com:

ClientIP	Bytes
192.168.1.3	17kB
192.168.1.4	12kB
192.168.1.2	11kB
192.168.1.5	4kB
192.168.1.6	2kB

www.server2.com:

ClientIP	Bytes
192.168.1.1	9kB
192.168.1.3	7kB
192.168.1.2	2kB
192.168.1.6	1kB
192.168.1.7	1kB

www.server3.com:

ClientIP	Bytes
192.168.1.1	19kB
192.168.1.4	15kB
192.168.1.3	12kB
192.168.1.5	5kB
192.168.1.7	2kB

Status of the Query Processing:

$min\text{-}k = 28$

Candidates:

Id	worstscore	bestscore
192.168.1.3	24	39
192.168.1.1	28	39
192.168.1.4	27	34
192.168.1.2	11	33

Query Processing Example (NRA)

www.server1.com:

ClientIP	Bytes
192.168.1.3	17kB
192.168.1.4	12kB
192.168.1.2	11kB
192.168.1.5	4kB
192.168.1.6	2kB

www.server2.com:

ClientIP	Bytes
192.168.1.1	9kB
192.168.1.3	7kB
192.168.1.2	2kB
192.168.1.6	1kB
192.168.1.7	1kB

www.server3.com:

ClientIP	Bytes
192.168.1.1	19kB
192.168.1.4	15kB
192.168.1.3	12kB
192.168.1.5	5kB
192.168.1.7	2kB

Status of the Query Processing:

$min\text{-}k = 28$

Candidates:

Id	worstscore	bestscore
192.168.1.3	24	39
192.168.1.1	28	39
192.168.1.4	27	29
192.168.1.2	13	28

Query Processing Example (NRA)

www.server1.com:

ClientIP	Bytes
192.168.1.3	17kB
192.168.1.4	12kB
192.168.1.2	11kB
192.168.1.5	4kB
192.168.1.6	2kB

www.server2.com:

ClientIP	Bytes
192.168.1.1	9kB
192.168.1.3	7kB
192.168.1.2	2kB
192.168.1.6	1kB
192.168.1.7	1kB

www.server3.com:

ClientIP	Bytes
192.168.1.1	19kB
192.168.1.4	15kB
192.168.1.3	12kB
192.168.1.5	5kB
192.168.1.7	2kB

Status of the Query Processing:

$min\text{-}k = 36$

Candidates:

Id	worstscore	bestscore
192.168.1.3	36	36
192.168.1.1	28	39
192.168.1.4	27	29
192.168.1.2	13	25

Query Processing Example (NRA)

www.server1.com:

ClientIP	Bytes
192.168.1.3	17kB
192.168.1.4	12kB
192.168.1.2	11kB
192.168.1.5	4kB
192.168.1.6	2kB

www.server2.com:

ClientIP	Bytes
192.168.1.1	9kB
192.168.1.3	7kB
192.168.1.2	2kB
192.168.1.6	1kB
192.168.1.7	1kB

www.server3.com:

ClientIP	Bytes
192.168.1.1	19kB
192.168.1.4	15kB
192.168.1.3	12kB
192.168.1.5	5kB
192.168.1.7	2kB

Status of the Query Processing:

$min\text{-}k = 36$

Candidates:

Id	worstscore	bestscore
192.168.1.3	36	36
192.168.1.1	28	32
192.168.1.4	27	29
192.168.1.2	13	25
192.168.1.5	4	18

Distributed Top-k

How to deal with distributed index-lists?

Problem: Number of round-trips
Solution: Batching.....?
Batch size??

Three Phase Uniform Threshold Algorithm (TPUT) [Cao&Wang, PODC 2004]

Phase 1

Fetch k **best entries** (d, s_j) from each of $P_1, .., P_m$ and aggregate $(\sum_{j=1..m} s_j(d))$ at P_0

Phase 2

Ask each of $P_1, .., P_m$ for **all entries with** $s_j > \mathit{min\text{-}k}/m$ and aggregate results in P_0.

Phase 3

Fetch missing scores for all candidates by random lookups at $P_1, .., P_m$

KLEE [Michel&Triantafillou&Weikum, VLDB 2005]

TPUT: Key Observations:

- If *min-k* is small TPUT retrieves a lot of data in Phase 2 $\rightarrow$ high network traffic
- Random accesses $\rightarrow$ high per-peer load

KLEE:

- Different philosophy: approximate answers.
- But far beyond TPUT without the third Phase!
- Efficiency:
  - Reduce (docId, score)-pair transfers
  - No random accesses at each peer
- Build on two novel data structures:
  - The **HistogramBlooms** structure
  - The **Candidate List Filter** structure

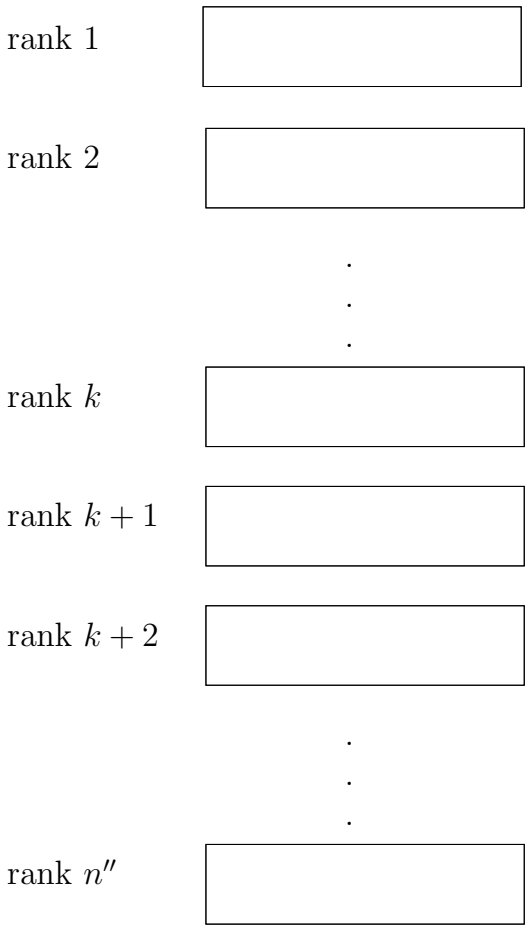
Probabilistic Guarantees

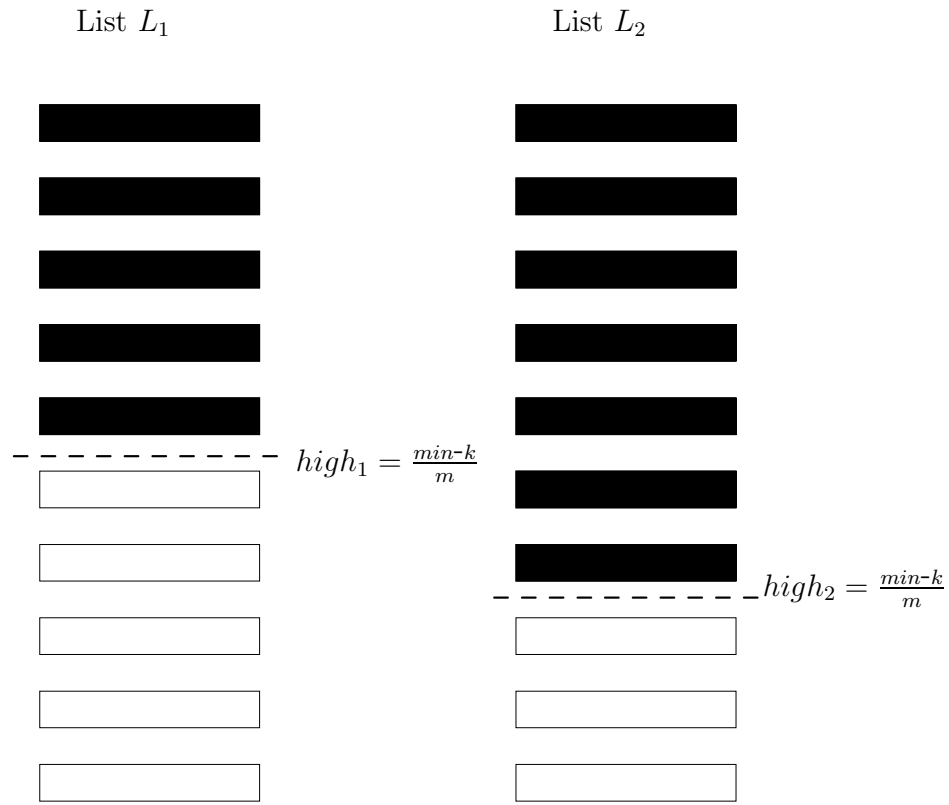
- TPUT is exact, i.e. it calculates the true top-k result.
- KLEE is approximate by design.

Assuming perfect compression (no false positives in the Bloom filters), both approximate algorithms have the same recall performance.

Consider the state of the algorithms after phase 2. What can we say about the quality of the current ranking compared to the true ranking?

State of the Algorithm after Phase 2





Guarantees

Assumptions

- The item currently at rank $k+1$ has the highest chance to get into the top-k result
- We do not distinguish between index lists (assuming the same score distribution in the tails).

An upper bound for missing a top-k item

We can derive an upper bound p_{upper} for the probability p_{miss} that denotes how likely it is to miss a true top-k item due to missing scores.

Expected Relative Recall

$$P[recall = r/k] = \binom{k}{r}(1 - p_{miss})^r p_{miss}^{(k-r)}$$

With

$$E[recall] \geq 1 - p_{upper}$$

We can furthermore derive Chernoff-Hoeffding bounds:

Theorem

For all $\psi > 0$ we have

$$P[|recall - E[recall]| > \psi] \leq 2e^{-2k\psi^2}$$

Optimizations

Adaptive Scan Depths

Adapt the *min-k* threshold in phase 2 to decrease network load.

Hierarchical Query Plans

Algorithms naturally support hierarchical query execution. Gives room for optimization techniques.

Index-list Sampling

Process only the most "valuable" index lists.

Conclusion

- Approximate Top-k: KLEE is algorithm of choice
- Probabilistic guarantees for result quality
- Three different ways to optimize distributed top-k queries
- Experiments show benefits in query response time and network traffic

Qu'est-ce que je fais aujourd'hui?

- Search in high-dimensional data
- Exploiting social networks: Tag prediction. Understanding Web 2.0 communities
- Stream processing
- Data/Metadata management for the environmental sciences: www.swiss-experiment.ch

Danke
Thank you
Epharisto
Merci
Obrigado