Top-K Aggregation Queries in Large-Scale Distributed Systems[*]

[PhD Thesis Summary]

Sebastian Michel
Max-Planck-Institut für Informatik, Saarbrücken, Germany
(Current: Ecole Polytechnique Fédérale de Lausanne, Switzerland)

Abstract: Distributed top-k query processing has become an essential functionality in a large number of emerging application classes like Internet traffic monitoring and Peer-to-Peer Web search. This work addresses efficient algorithms for distributed top-k queries in wide-area networks where the index lists for the attribute values (or text terms) of a query are distributed across a number of data peers.

1 Introduction

The success and growth of the Web and the Internet is spurring the development of an ever increasing number of interesting application classes, from Internet-scale monitoring, to aggregation queries in sensor networks, and to peer-to-peer Web searching. Internet traffic monitoring is crucial for understanding the nature of modern applications' load characteristics such as P2P file sharing, news feeds, or Blogs, and uses network instrumentation at different levels and time scales. As the underlying routers and other components of the observatory infrastructure are highly distributed, analyzing the logged data often requires distributed aggregation and iceberg queries (i.e., top-k computations over aggregated traffic measures). Similar requirements arise for detecting traffic anomalies such as network intrusions or denial-of-service attacks. Sensor networks are gaining great importance for monitoring environmental data such as water quality measures in rivers or other measurements of the physical world. Here, too, evaluating the data naturally leads to distributed aggregation queries where one is often interested only in the top-k query results, e.g., the top water streams with the highest nitrate concentration. P2P Web search is an emerging alternative to centralized search engines that bear various intriguing potentials: lower susceptibility to search engine spam and manipulation, exploitation of behavior and recommendations of users and entire user communities implicit in bookmarks, query logs, and click streams, and collaborative search for advanced expert queries. In such a setting, queries would combine page scoring information from several peers that maintain different index lists. As in standard Web search, users often look only at the top-10 results. From our point of view, the common key feature of all such applications is that the data are distributed over a number of nodes at large scale and in an ad-hoc manner, and that this

[*]This paper is a summary of the PhD thesis "Top-K Aggregation Queries in Large-Scale Distributed Systems", by Sebastian Michel (Max-Planck-Institut für Informatik), Universität des Saarlandes, Saarbrücken, Germany, 2007.

data must be collected, and some aggregation function be applied, with the desired goal being the identification of the k most relevant/interesting data items. We focus on efficient top-k query algorithms in distributed environments.

1.1 Problem Statement and Computational Model

We consider a distributed system with N peers, P_j, $j = 1, ..., N$, that are connected, e.g., by a distributed hash table or some overlay network. Data items are either documents such as Web pages or structured data items such as movie descriptions. Each data item has associated with it a set of descriptors, text terms or attribute values, and there is a precomputed score for each pair of data item and descriptor. The inverted index list for one descriptor is the list of data items in which the descriptor appears sorted in descending order of scores. These index lists are the distribution granularity of the distributed system. Each index list is assigned to one peer (or, if we wish to replicate it, to multiple peers).

The overall goal is to efficiently find the top-k items (documents) or, in case of approximate algorithms, come as close as possible to the true top-k results. We measure the quality of the approximate result by the fraction of documents in the approximate top-k result that are also in the true top-k result, i.e. the relative recall.

1.2 Contribution

- We present the KLEE algorithmic framework [MTW05a] as a fundamental building block towards efficient top-k query processing in distributed systems.
- We present techniques to model value distributions and show how these models can be used to reason about parameter values that play an important role in the overall performance of KLEE.
- We present the GRASS algorithmic framework. The GRASS algorithms come with three different kinds of optimization techniques [NBM$^+$08]: we introduce a technique to efficiently leverage the knowledge of the input data characteristics to tune thresholds that are of fundamental importance. We show how hierarchical query plans can be generated using the aforementioned cost model to build optimal query execution plans that drastically increase the overall performance. We introduce a method to select a sample of input data sources that still provides reasonably accurate results but can be, at the same time, more efficiently handled. All these techniques result in a significantly increased overall performance.
- We present probabilistic guarantees for the aforementioned algorithms.
- Moreover, we address the issue of building a highly scalable search engine. We have developed Minerva∞ [MTW05b], a scalable and efficient Peer-to-Peer Web search engine. The distinguishing feature of Minerva∞ is the high distribution both in the data and computational dimensions. The key idea is to give up the nodes' autonomy and distribute each index list over multiple peers, as a key step towards a system with unlimited scalability. We expect that nodes will autonomously crawl the web, discovering *documents* and computing *scores* of documents, with each score reflecting a document's importance with respect to *terms* of interest.

2 Related Work

Top-k query processing has received much attention in a variety of settings such as similarity search on multimedia data [CGM04, CGM04, Fa99, FLN03, GBK00, BGRS99, NCS$^+$01, dVMNK02], ranked retrieval on text and semi-structured documents in digital libraries and on the Web [AdKM01, TWS04, BJRS03], spatial data analysis [BBK01, CP02], network and stream monitoring [BO03, KOT04, CW04] collaborative recommendation and preference queries on e-commerce product catalogs [YPM03, MBG04, BGM02, GBK01], and ranking of SQL-style query results on structured data sources in general [ACDG03, CDHW04, BCG02]. [BCG02] addresses the mapping of top-k queries into range queries that can be handles by the query optimizer in a conventional RDBMS.

The first distributed TA-style algorithm has been presented in [BGM02, MBG04]. The emphasis of that work was on top-k queries over Internet data sources for recommendation services (e.g., restaurant ratings, street finders). Because of functional limitations and specific costs of data sources, the approach used a hybrid algorithm that allowed both sorted and random access but tried to avoid random accesses. Scheduling strategies for random accesses to resolve expensive predicates were addressed also in [CwH02]. In our widely distributed setting, none of these scheduling methods are relevant as they still incur an unbounded number of message rounds. The method in [SMwW$^+$03] addresses P2P-style distributed top-k queries but considers only the case of two index lists distributed over two peers. Its key idea is to allow the two cohort peers to directly exchange score and candidate information rather than communicating only via the query initiator. Unfortunately, it is unclear and left as an open issue how to generalize to more than two peers.

In contrast, state-of-the-art algorithms for distributed top-k aggregation use a fixed number of communication rounds to bound latency and aim to minimize the total network bandwidth consumption. The first algorithm in this family was the *TPUT* (Three-Phase Uniform Threshold) algorithm [CW04], in which a query coordinator, typically the network node which initiates the query, executes a three-phase distributed threshold algorithm. We will give a more detailed description of the algorithm in the following section. *TPAT* [YLW$^+$05] is a modification of TPUT where the threshold, that is the same for all index lists, is adapted to the specifics of the value distributions; however, the authors state that their solution may incur infeasible computational cost. We will also consider the issue of adaptive thresholds and introduce an efficient way to calculate them.

2.1 The Three Phase Uniform Threshold Algorithm (TPUT)

The main idea of TPUT [CW04] is to transform the top-k query into a range query where the range is determined via an estimation of the min-k value.

1. min-k **estimation phase:** The query initiator P_{init} retrieves the top k items from each of the input index-lists. Subsequently, P_{init} calculates the worstscore for all observed items and ranks them accordingly. The worstscore is the aggregation of the actually observed values. Similarly, the bestscore denotes the best possible score an item can achieve in the future. The worstscore of the item currently at rank k is denoted as min-k.

2. **Candidate retrieval phase:** Based on the $min\text{-}k$ estimation, TPUT sends the $min\text{-}k/m$ threshold to all involved peers that send back all $(itemId, score)$-pairs with $score \geq min\text{-}k/m$. This ensures that all candidates (potential members of the final top-k result) have been found, in at least one of the lists. After the $min\text{-}k$ value has been recalculated, TPUT throws away all items with $bestscore < min\text{-}k$.
3. **Missing scores lookup phase:** For all the remaining candidates, TPUT looks up the missing scores by sending to each peer P_i a list of the candidates that have not been seen in list L_i so far. P_{init} can now calculate the exact score for all candidates, i.e. the true top-k results have been identified.

Phase 1			**Phase 2**			**Phase 3**		
$\boldsymbol{(a,12)}$	$\boldsymbol{(b,8)}$	$\boldsymbol{(a,17)}$	$\boldsymbol{(a,12)}$	$\boldsymbol{(b,8)}$	$\boldsymbol{(a,17)}$	$\boldsymbol{(a,12)}$	$\boldsymbol{(b,8)}$	$\boldsymbol{(a,17)}$
$\boldsymbol{(b,10)}$	$\boldsymbol{(c,7)}$	$\boldsymbol{(z,13)}$	$\boldsymbol{(b,10)}$	$\boldsymbol{(c,7)}$	$\boldsymbol{(z,13)}$	$\boldsymbol{(b,10)}$	$\boldsymbol{(c,7)}$	$\boldsymbol{(z,13)}$
$(c,8)$	$(e,6)$	$(e,11)$	$\boldsymbol{(c,8)}$	$\boldsymbol{(e,6)}$	$\boldsymbol{(e,11)}$	$\boldsymbol{(c,8)}$	$\boldsymbol{(e,6)}$	$\boldsymbol{(e,11)}$
$(d,6)$	$(z,4)$	$(f,10)$	$\boldsymbol{(d,6)}$	$(z,4)$	$\boldsymbol{(f,10)}$	$\boldsymbol{(d,6)}$	$\boldsymbol{(z,4)}$	$\boldsymbol{(f,10)}$
$(e,3)$	$(m,2)$	$(c,6)$	$(e,3)$	$(m,2)$	$\boldsymbol{(c,6)}$	$\boldsymbol{(e,3)}$	$(m,2)$	$\boldsymbol{(c,6)}$
$(h,3)$	$(g,2)$	$(r,5)$	$(h,3)$	$(g,2)$	$(r,5)$	$(h,3)$	$(g,2)$	$(r,5)$
$(f,2)$	$(o,1)$	$(b,5)$	$(f,2)$	$(o,1)$	$(b,5)$	$(f,2)$	$(o,1)$	$\boldsymbol{(b,5)}$

Table 1: Sample TPUT execution for a top-2 query: Phase 1 (left): Retrieve the top-2 items from each list. Phase 2 (middle): Retrieve all items with score above $min\text{-}k/3$ = 6. Phase 3 (right): Retrieve missing score via random lookups.

3 The KLEE Algorithmic Framework

TPUT ensures a small query response times using a small, fixed number of (only three) communication phases. We also adopt the requirement for a small number of communication phases. However, KLEE goes far beyond. The salient features and novel contributions of KLEE are the following:

- KLEE comes with two flavors, one involving only two and one involving three communication phases. It recognizes that the number of communication phases is only one aspect of guaranteeing short response times, which, in turn, is only one aspect of overall efficiency. In particular, as limited network and IO bandwidth appear to be key contributors to response times, KLEE ensures that significantly smaller messages are exchanged and that random IOs at participating peers are avoided, resulting in strong gains in response time and network bandwidth and lighter peer loads compared to TPUT.
- KLEE is the first to make a strong case for approximate top-k algorithms for wide-area networks, showing how significant performance benefits can be enjoyed, at only small penalties in result quality.
- KLEE provides a flexible framework for top-k algorithms, allowing for trading-off efficiency versus result quality and bandwidth savings versus the number of communication phases.
- We have implemented KLEE and a number of competing algorithms and conducted comprehensive experimental performance evaluation using real-world and synthetic data, which shows the consistent superiority of KLEE over its competitors.

- KLEE is equipped with various fine-tuning parameters and we provide a discussion of how these can be automatically adjusted to underlying data and system characteristics.

The proposed approach is based on having a per-query coordinator peer and a set of cohort peers. In our setting, the coordinating peer is the peer where the query was initiated, P_{init}. The cohort peers, are the peers storing the index lists, based on which the document scores will be computed. The algorithm is structured to proceed in a number of phases, with each phase consisting of a round-trip communication between the coordinator and the cohorts. In general, in each phase, the coordinator requests and receives from each peer a portion of the peer's local index information, which permits the coordinator to run a top-k algorithm (such as the TA algorithm or variants) based on the collected information about the peers' index lists.

3.1 The HistogramBlooms Structure

In KLEE, each peer maintains a set of statistical metadata describing its index list. In particular, histogram-based information is maintained to describe the distribution of scores in the index list. The range of possible score values cover the range $(0, 1]$. For simplicity, we assume that peer histograms are equi-width, consisting of n cells, each cell being responsible for $(1/n)th$ of the score range. It would be straightforward to employ other forms of histograms. Associated with each cell i, each peer maintains the following information

- The lower and upper values, $lb[i]$, $ub[i]$, respectively, defining the range of scores being covered by this cell,
- The value of $freq[i]$, defining the number of document IDs whose scores in the peer's index list fall within $lb[i]$ and $ub[i]$,
- The average score, $avg[i]$, computed over all scores in the cell, and
- A synopsis of the document IDs whose scores fall in this cell, $filter[i]$. In particular, this compact representation is constructed using Bloom filters [Bl70].

In the first phase, at the coordinator's request, each cohort peer replies with its local top-k list, and a fraction of its HistogramBlooms data structure. The coordinator then can address the missing-scores problem as follows: for every peer P_i that has not reported a score for docID, using the Bloom-filter cell summaries of P_i and the hash functions, it can find to which histogram cell of peer P_i the docID belongs say c, (by simply testing for membership of docID in the filters of each cell, and stopping when a test is successful). Then, it can use the average score associated with that histogram cell, $avg[c]$, to replace the missing score of P_i for docID.

With the additional knowledge of the documents' scores, the query initiator is able to compute a tighter $min\text{-}k/m$ score bound which leads to a decreased network consumption in the second communication phase as less documents qualify for the range query.

KLEE comes in addition use of an optional filtering phase for the data transmission in the second phase (cf., [MTW05a]).

4 Probabilistic Guarantees

TPUT is exact, i.e. it calculates the true top-k result. KLEE [MTW05a] is an approximate version of TPUT that does not employ the third phase and, in addition, uses compression techniques based on Bloom filters and a technique to filter out unpromising candidates.

For now we assume that the compression techniques applied in KLEE are perfect, i.e. do not cause any false positives. It is straightforward to configure Bloom filters to provide this error-free behavior with high probability [Bl70]. So, we actually consider the approximate version of TPUT (without the third phase) and disregard KLEE's optimization techniques.

We consider the state of the algorithm after the second phase. All items with bestscore $< min\text{-}k$ have been pruned away. This is the standard pruning technique that does not introduce errors. The error in the relative recall at this stage of the algorithm is caused by the ordering of the items since this ordering is based on incomplete information (i.e., not fully evaluated candidate items). We assume that the item that is currently at rank $k + 1$ has the highest chance to get into the top-k result. In general, an item at rank $> k + 1$ can have a higher chance to get into the top-k than the item rank $k + 1$ if it has been seen in fewer index lists than the item at rank $k + 1$ and thus may achieve a higher final score. In our scenario it does not make sense to distinguish between the different index lists since for all of them the algorithm reads down to $min\text{-}k/m$ and we do not consider the score distributions in the tails of the index lists. Since we try to derive a general probabilistic guarantee, we cannot treat items individually. In this setting, assuming that the item at rank $k + 1$ has the highest chance to get into the top-k is a meaningful assumption.

With this assumption we have derived an upper bound for the probability that the item at rank-$k+1$ will get into the final top-k result, hence can derive the expected result accuracy [Mi07].

5 Optimizing Top-k Queries

We consider the optimization of distributed top-k queries in wide-area networks: We present GRASS, an algorithmic framework [NBM$^+$08] that consists of three optimization techniques.

- we introduce a technique to efficiently leverage the knowledge of the input data characteristics to tune score thresholds that are of fundamental importance. The basic KLEE method and related algorithms transform the top-k retrieval problem into range-queries where the ranges are determined using uniform score thresholds. We propose the usage of non-uniform thresholds, and present an efficient optimization algorithm to adapt the threshold to the index-lists score distribution characteristics.
- we show how hierarchical query plans can be generated using the aforementioned cost model to build optimal query execution plans that drastically increase the overall performance. Consider, for example, a query with one very large and several small input lists residing on different peers. It would be better to perform the top-k query at the peer with the large list, have the small peers ship their items to the large peer, and only send the final result to the query initiator. We use a dynamic programming approach that considers all possible query plans and chooses the cheapest plan w.r.t. our cost model.

- we introduce a sampling method to select a subset of the input data sources that still provides accurate results but can be, at the same time, more efficiently handled. We have performed experiments on real Web data that show the benefits of distributed top-k query optimization both in network resource consumption and query response time.

6 Minerva Infinity

An architectural alternative to the computational model that underlies our algorithms is to distribute each index list over multiple peers, as a key step towards a system with unlimited scalability. We expect that nodes will autonomously crawl the web, discovering documents and computing scores of documents, with each score reflecting a document's importance with respect to terms of interest. This results in index lists, one for each term, containing relevant documents and their scores for a term. In a succeeding step, each peer distributes its set of *(docId, score, term)-triplets* across the participating peers. One way of doing so is to use a standard order-preserving hash function that assigns each triplet to a node based on the score's hash-value plus a term-specific offset. While this obviously distributes the triplets over the peers it will create a load imbalance because of the skewed (Zipf-like) score distribution typically observed in real-world index lists. To overcome this problem, we have developed a more sophisticated hash function that distributes index lists over the participating peers in a load-balancing and, at the same time, order-preserving way. But even with such a hash function it is infeasible to distribute a single index list over all peers, since this would cause a gigantic communication overhead as all peers would have to be contacted in order to retrieve the required information. To overcome this problem, we restrict the placement of the *(docid, score, term)-triplets* for a particular term to a subset of all peers. These small networks, called term index networks (TIN), help to limit the number of peers contacted during retrieval. In general, TINs can form separate overlay networks, but for simplicity we model a TIN simply as a (circular) doubly-linked list. The top-k query processing proceeds in rounds, in which a coordinator peer retrieves batches of (docId, score, term)-triplets from the nodes that are part of the query-term specific TINs. We believe that our design choices are a big step towards a scalable P2P search engine. The Minerva∞ [MTW05b] architecture has been implemented, and performance experiments have been conducted.

7 Conclusion

We have considered distributed top-k algorithms, where the index lists for the query attributes are spread across multiple peers. We have presented the KLEE framework for distributed top-k query processing. KLEE's salient features set it apart from related work in several ways. First, KLEE makes for the first time a strong case for approximate top-k algorithms in widely distributed environments. Second, KLEE allows the trading-off of result quality vs performance. KLEE even allows for trading-off between bandwidth vs the number of communication phases. KLEE achieves great performance gains in network bandwidth, query response times, and local peer load, and high quality results. Moreover, we have considered means to model score distributions and how these score models can be used to reason about parameter values that play an important role in the overall performance of KLEE. We have presented the family of GRASS algorithms that use these score models for optimizing the query execution. GRASS shows significant performance gains, in particular for queries that involve many data sources. We have derived probabilistic guarantees for the algorithms presented, to show both analytically and by a comprehensive experimental study their suitability for various application classes. Furthermore, we have presented Minerva∞, a novel architecture for a peer-to-peer web search engine. The key distinguishing feature of Minerva∞ is its high-levels of distribution for both data and processing.

8 Acknowledgment

I would like to express my gratitude to all those who gave me the possibility to complete my PhD thesis. I am deeply indebted to my supervisor Prof. Dr.-Ing. Gerhard Weikum, Max-Planck-Institut für Informatik, Saarbrücken. I have been very lucky being able to work with him and a number of other very nice and highly talented people during my PhD. In particular notable for their manifold contributions to this thesis, Prof. Dr. Peter Triantafillou, University of Patras, Greece, and Dr. Thomas Neumann, Max-Planck-Institut für Informatik, Saarbrücken.

References

[ACDG03] Agrawal, S., Chaudhuri, S., Das, G., and Gionis, A.: Automated ranking of database query results. In: *1st Biennial Conference on Innovative Data Systems Research (CIDR), Asilomar, CA, USA, January 5-8, 2003*. 2003.

[AdKM01] Anh, V. N., de Kretser, O., and Moffat, A.: Vector-space ranking with effective early termination. In: Croft, W. B., Harper, D. J., Kraft, D. H., and Zobel, J. (Hrsg.), *SIGIR 2001: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, September 9-13, 2001, New Orleans, Louisiana, USA*. S. 35–42. ACM. 2001.

[BBK01] Böhm, C., Berchtold, S., and Keim, D. A.: Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Comput. Surv.* 33(3):322–373. 2001.

[BCG02] Bruno, N., Chaudhuri, S., and Gravano, L.: Top-k selection queries over relational databases: Mapping strategies and performance evaluation. *ACM Trans. Database Syst.* 27(2):153–187. 2002.

[BGM02] Bruno, N., Gravano, L., and Marian, A.: Evaluating top-k queries over web-accessible databases. In: *Proceedings of the 18th International Conference on Data Engineering, ICDE, 26 February - 1 March 2002, San Jose, CA*. S. 369–. IEEE Computer Society. 2002.

[BGRS99] Beyer, K. S., Goldstein, J., Ramakrishnan, R., and Shaft, U.: When is "nearest neighbor" meaningful? In: Beeri, C. and Buneman, P. (Hrsg.), *Database Theory - ICDT '99, 7th International Conference, Jerusalem, Israel, January 10-12, 1999, Proceedings.* volume 1540 of *Lecture Notes in Computer Science*. S. 217–235. Springer. 1999.

[BJRS03] Bawa, M., Jr., R. J. B., Rajagopalan, S., and Shekita, E. J.: Make it fresh, make it quick: searching a network of personal webservers. In: Proceedings of the Twelfth International World Wide Web Conference, WWW2003, Budapest, Hungary, 20-24 May 2003. S. 577–586. 2003.

[Bl70] Bloom, B. H.: Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*. 13(7):422–426. 1970.

[BO03] Babcock, B. and Olston, C.: Distributed top-k monitoring. In: Halevy, A. Y., Ives, Z. G., and Doan, A. (Hrsg.), *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, San Diego, California, USA, June 9-12, 2003*. S. 28–39. ACM. 2003.

[CDHW04] Chaudhuri, S., Das, G., Hristidis, V., and Weikum, G.: Probabilistic ranking of database query results. In: Nascimento, M. A., Özsu, M. T., Kossmann, D., Miller, R. J., Blakeley, J. A., and Schiefer, K. B. (Hrsg.), *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, August 31 - September 3 2004*. S. 888–899. Morgan Kaufmann. 2004.

[CGM04] Chaudhuri, S., Gravano, L., and Marian, A.: Optimizing top-k selection queries over multimedia repositories. *IEEE Trans. Knowl. Data Eng.* 16(8):992–1009. 2004.

[CP02] Ciaccia, P. and Patella, M.: Searching in metric spaces with user-defined and approximate distances. *ACM Trans. Database Syst.* 27(4):398–437. 2002.

[CW04] Cao, P. and Wang, Z.: Efficient top-k query calculation in distributed networks. In: Chaudhuri, S. and Kutten, S. (Hrsg.), *Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing, PODC 2004, St. John's, Newfoundland, Canada, July 25-28, 2004*. S. 206–215. ACM. 2004.

[CwH02] Chang, K. C.-C. and won Hwang, S.: Minimal probing: supporting expensive predicates for top-k queries. In: Franklin, M. J., Moon, B., and Ailamaki, A. (Hrsg.), *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, June 3-6, 2002*. S. 346–357. ACM. 2002.

[dVMNK02] de Vries, A. P., Mamoulis, N., Nes, N., and Kersten, M. L.: Efficient k-nn search on vertically decomposed data. In: Franklin, M. J., Moon, B., and Ailamaki, A. (Hrsg.), *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, June 3-6, 2002*. S. 322–333. ACM. 2002.

[Fa99] Fagin, R.: Combining fuzzy information from multiple systems. *J. Comput. Syst. Sci.* 58(1):83–99. 1999.

[FLN03] Fagin, R., Lotem, A., and Naor, M.: Optimal aggregation algorithms for middleware. *J. Comput. Syst. Sci.* 66(4):614–656. 2003.

[GBK00] Güntzer, U., Balke, W.-T., and Kießling, W.: Optimizing multi-feature queries for image databases. In: Abbadi, A. E., Brodie, M. L., Chakravarthy, S., Dayal, U., Kamel, N., Schlageter, G., and Whang, K.-Y. (Hrsg.), *Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*. S. 419–428. Morgan Kaufmann. 2000.

[GBK01] Güntzer, U., Balke, W.-T., and Kießling, W.: Towards efficient multi-feature queries in heterogeneous environments. In: International Symposium on Information Technology (ITCC 2001), 2-4 April 2001, Las Vegas, NV, USA. S. 622–628. IEEE Computer Society. 2001.

[KOT04] Koudas, N., Ooi, B. C., Tan, K.-L., and 0003, R. Z.: Approximate nn queries on streams with guaranteed error/performance bounds. In: Nascimento, M. A., Özsu, M. T., Kossmann, D., Miller, R. J., Blakeley, J. A., and Schiefer, K. B. (Hrsg.), *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, August 31 - September 3 2004*. S. 804–815. Morgan Kaufmann. 2004.

[MBG04] Marian, A., Bruno, N., and Gravano, L.: Evaluating top-*k* queries over web-accessible databases. *ACM Trans. Database Syst.* 29(2):319–362. 2004.

[Mi07] Michel, S.: *Top-K Aggregation Queries in Large-Scale Distributed System*. PhD thesis. Universität des Saarlandes. Saarbrücken, Germany. 2007.

[MTW05a] Michel, S., Triantafillou, P., and Weikum, G.: Klee: A framework for distributed top-k query algorithms. In: Böhm, K., Jensen, C. S., Haas, L. M., Kersten, M. L., Larson, P.-Å., and Ooi, B. C. (Hrsg.), *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005*. S. 637–648. ACM. 2005.

[MTW05b] Michel, S., Triantafillou, P., and Weikum, G.: Minerva$_{\text{infinity}}$: A scalable efficient peer-to-peer search engine. In: Alonso, G. (Hrsg.), *Middleware*. volume 3790 of *Lecture Notes in Computer Science*. S. 60–81. Springer. 2005.

[NBM$^+$08] Neumann, T., Bender, M., Michel, S., Schenkel, R., Triantafillou, P., and Weikum, G.: Optimizing distributed top-k queries. In: Web Information Systems Engineering - WISE, 9th International Conference, Auckland, New Zealand, September 1-3. S. 337–349. 2008.

[NCS$^+$01] Natsev, A., Chang, Y.-C., Smith, J. R., Li, C.-S., and Vitter, J. S.: Supporting incremental join queries on ranked inputs. In: Apers, P. M. G., Atzeni, P., Ceri, S., Paraboschi, S., Ramamohanarao, K., and Snodgrass, R. T. (Hrsg.), *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy*. S. 281–290. Morgan Kaufmann. 2001.

[SMwW$^+$03] Suel, T., Mathur, C., wen Wu, J., Zhang, J., Delis, A., Kharrazi, M., Long, X., and Shanmugasundaram, K.: Odissea: A peer-to-peer architecture for scalable web search and information retrieval. In: Christophides, V. and Freire, J. (Hrsg.), *WebDB*. S. 67–72. 2003.

[TWS04] Theobald, M., Weikum, G., and Schenkel, R.: Top-k query evaluation with probabilistic guarantees. In: Nascimento, M. A., Özsu, M. T., Kossmann, D., Miller, R. J., Blakeley, J. A., and Schiefer, K. B. (Hrsg.), *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, August 31 - September 3 2004*. S. 648–659. Morgan Kaufmann. 2004.

[YLW$^+$05] Yu, H., Li, H.-G., Wu, P., Agrawal, D., and Abbadi, A. E.: Efficient processing of distributed top- queries. In: Andersen, K. V., Debenham, J. K., and Wagner, R. (Hrsg.), *Database and Expert Systems Applications, 16th International Conference, DEXA 2005, Copenhagen, Denmark, August 22-26, 2005, Proceedings*. volume 3588 of *Lecture Notes in Computer Science*. S. 65–74. Springer. 2005.

[YPM03] Yu, C. T., Philip, G., and Meng, W.: Distributed top-n query processing with possibly uncooperative local systems. In: Freytag, J. C., Lockemann, P. C., Abiteboul, S., Carey, M. J., Selinger, P. G., and Heuer, A. (Hrsg.), *VLDB 2003, Proceedings of 29th International Conference on Very Large Data Bases, September 9-12, 2003, Berlin, Germany*. S. 117–128. Morgan Kaufmann. 2003.