Subscription Covering for Relevance-based Filtering in Content-Based Publish/Subscribe Systems

Kaiwen Zhang[1,4], Vinod Muthusamy[2], Mohammad Sadoghi[3], Hans-Arno Jacobsen
[1]Technische Universität München
[2]IBM T.J. Watson Research Center
[3]Purdue University
[4]University of Toronto

***Abstract*—Large-scale applications require a scalable data dissemination service with advanced filtering capabilities. We propose the use of a content-based publish/subscribe system with support for top-k filtering in the context of such applications. We focus on the problem of top-k subscription filtering, where a publication is delivered only to the k highest scoring subscribers. The naive approach to perform filtering early at the publisher edge works only if complete knowledge of the subscriptions is available, which is not compatible with the well-established covering optimization in scalable content-based publish/subscribe systems. We propose an efficient rank-cover technique to reconcile top-k subscription filtering with covering. We extend the covering model to support top-k and describe a novel algorithm for forwarding subscriptions to publishers while maintaining correctness. Finally, we compare our solutions to a baseline covering system. In a typical setting, our optimized solution is scalable and provides over 81% of the covering benefit.**

I. Introduction

The volume of data is increasing exponentially, with some estimating that over 2.5 billion gigabytes of information is generated daily [1]. Given its sheer size, new challenges arise for distributed content dissemination. Therefore, the ability for producers to deliver pertinent data only to selected and interested consumers is an important requirement which exists in many domains.

For instance, in targeted marketing, the merchants wish to limit the size of their advertising audience based on their social network profile and context such as location. For example, there could be a finite number of promotions to be delivered only to individuals who are most likely to be interested in the offer or redeem the coupon. Selecting the matching individuals may be based on a ranking based on the *relevance* of the clients' profile.

The above scenario requires a network of interconnected consumers (i.e., subscribers) and producers of data (i.e., publishers) with complex filtering requirements. Given its inherently distributed nature, there is a need to establish a decentralized architecture to cope with the data volume. The content-based pub/sub model, known for its scalability and decoupled nature, is a logical candidate for a distributed data dissemination substrate [2]–[14]. Furthermore, it has been shown that that social networks can be built on top of pub/sub primitives [15], [16].

Although content-based pub/sub systems already offer fine-grained filtering capabilities, there is a need to explicitly limit publication deliveries (*top-k filtering*) in order to restrict network overhead and information overload. In some applications, a subscriber may wish to control the amount of publications it receives (*consumer-centric* filtering), while in others, a publisher may want to limit the number of subscribers each publication is delivered to (*producer-centric* filtering). The targeted marketing scenario calls for the latter capability and is the problem addressed in this paper. We also remark that producer-centric top-k filtering is a better fit for pub/sub, since it is processed as a stateless operation which does not store publications, which is what pub/sub systems are innately suited for. On the other hand, consumer-centric top-k filtering is a stateful operation which requires the pub/sub system to be extended to deal with window semantics, which introduces publication buffering and consistency issues, as described in our prior work [10].

More specifically, this paper focuses on delivering each publication to its k highest ranked matching subscriptions over a pub/sub broker overlay. Subscription rankings are based on the relevance of the content of the publication to the associated subscribers.

There is a requirement for pub/sub systems to employ optimized routing protocols and efficient filtering mechanisms for disseminating information flowing from data producers to data consumers. A well-known routing optimization prevalent in content-based pub/sub systems is covering [17]. This protocol reduces the propagation and management of subscriptions that are forwarded to brokers (i.e., nodes) in the pub/sub network. Covering optimization algorithms are compatible only with simple subscription matching and do not extend trivially to more expressive semantics such as top-k filtering.

In general, without the covering protocol, every edge broker to which a publisher is connected to maintains global knowledge of all subscriptions in the system, thereby making top-k subscription filtering a centralized task since the rank of all subscriptions can be computed locally. But once subscription covering is assumed (as is commonly the case in pub/sub systems), the edge broker has only partial knowledge of existing subscriptions. Top-k filtering then becomes a challenging and important problem in that setting.

In this paper, we make the following contributions:

1) Formalize top-k subscription filtering semantics based on relevance on the publication to each subscriber.
2) Develop a novel rank-cover algorithm (RankCoverK) which leverages a partially ordered set (poset) of subscriptions to enable relevance-based top-k subscription filtering while supporting covering.
3) Introduce an ancestor counting optimization (AncestorCoverK) to further reduce the size of the covering poset produced by RankCoverK and the resulting subscription traffic.
4) Conduct a scalability analysis of our algorithms based on an open-source pub/sub system.

II. Related Work

The most widely adopted database top-k processing model [18], [19] differs from our proposed top-k model in an important respect: our top-k model solves the reverse problem. In the database context, top-k querying means finding the most relevant tuples (events) for a given query (subscription). But in our pub/sub abstraction, matching means finding the relevant subscriptions (queries) for a given event (tuple).

An important aspect of pub/sub top-k matching is to explore and identify a set of plausible top-k semantics. Unlike in the database context, formalizing top-k semantics in pub/sub is more involved and not limited to a single interpretation [20]–[22]. The most widely used pub/sub top-k semantics is defined with respect to subscribers, i.e., consumer-centric semantics, in which the subscription language is extended (with a scoring function) in order to rank each incoming publication (over a time- or count-based sliding window); thus, delivering only the top-k matched publications to each subscriber [20]–[22].

Alternatively, the top-k semantics can be defined with respect to a publisher, i.e., producer-centric semantics, which extends the publication language for ranking subscribers and delivering publications only to the top-k matched subscribers [6], [23]. Producer-centric semantics is suitable for targeted advertisement (e.g., targeting a specific demographic group) and diversified advertisement (e.g., reaching out to most eligible members within each demographic group). Finally, hybrid semantics can be foreseen such that both subscribers and publishers have control on how data is received and disseminated, respectively.

Recent works argue for the importance of top-k matching based on the relevance of subscriptions (i.e., producer-centric) using a non-monotonic and dynamic scoring [12], but without paying attention to the covering routing techniques employed by pub/sub systems which is the focus of our work. A window-based top-k matching solution that also considered covering was studied in [10], but unlike our present work, their focus was on consumer-centric filtering.

Other notable top-k research directions in pub/sub are as follows. To reduce memory footprint, a probabilistic model with a bounded-error was developed in order to determine whether a given publication will eventually fall in top-k results of at least one subscriber [20]. The problems of how to formulate the subscriber's interest and ranking publications were studied in [21]. To further improve the relevance of top-k publication results (e.g., improving the diversity), different top-k measures such as quantitative approaches (based on scoring functions) and qualitative approaches (based on explicit binary relation preferences) were explored in [22]. More recently, the problem of top-k diversification was also studied in the context of finding the most diverse set of text documents for each subscriber over data streams [24]. Furthermore, the problem of computing top-k results under spatial constraints were studied in [13], [14]. Unlike these approaches, our work focuses on efficient subscription covering which supports distributed top-k computation in order to reduce overall network traffic.

III. Semantics and naive solution

In this section, we first revise the conventional definition of subscription covering used in the context of advertisement-based routing with content-based matching. We then formalize a model for top-k subscription filtering in pub/sub using relevance scoring. Second, we describe how top-k subscription filtering semantics are propagated through the system by piggybacking top-k parameters via advertisements. Finally, we demonstrate how covering, in conjunction with the proposed top-k model, is incompatible with a naive solution and is thus a non-trivial challenge to solve.

A. Pub/Sub model with subscription covering

In a content-based matching model, subscriptions consist of a conjunction of predicates of the form $(attribute, operator, value)$. These predicates are then matched against the attribute-value pairs ($attribute$, $value$) of incoming publications. A publication p is considered *matching* for a subscription s if all its predicates are satisfied by p. In that case, p must be delivered to s. We employ advertisement-based routing, where publishers must first send advertisements which are flooded through the broker overlay. Subscriptions are routed through the network by matching them against advertisements and forwarding them to *upstream* brokers (towards the corresponding publishers). Subsequently, publishers are allowed to publish publications only within the space they advertised.

Subscription covering is a routing optimization which reduces the volume of subscriptions to be propagated through the pub/sub system. A set of incoming subscriptions $\mathbb{S}$ is then processed as shown in Figure 1. First, the subs are matched against all existing advertisements $\mathbb{A}$ to determine which publishers can publish potentially matching publications.

Second, we determine the covering relationships between existing subscriptions and the new ones in $\mathbb{S}$. A subscription s_1 covers subscription s_2 if all publications that match s_2 also match s_1. This can be determined by inspecting the predicates of the two subscriptions. For instance, suppose the predicates of s_1 are $x \in [1, 10], y \in [1, 10]$, and s_2 are $x \in [1, 5], y \in [1, 10]$. In this situation, s_1 covers s_2.

After applying covering, we obtain a subset C of $\mathbb{S}$ with subscriptions which are *not* covered. C is then forwarded towards the sources of advertisements found in A. Note that any subscription in C which has previously been sent is not forwarded again.

Consider the example in Figure 2, where a publisher is connected to broker B_1 and the subscribers connected to brokers B_2 and B_3. Suppose that the subscribers at broker B_2 issue the subscriptions s_1 to s_7 of the form $x \in [a, b], y \in [c, d]$, which are listed in Table I. We can build a covering partially ordered set (poset) for B_2, as depicted in Figure 3. Higher level subscriptions cover their descendants. In this example, s_1 covers subscriptions s_2 to s_7. Therefore, applying the covering optimization allows broker B_2 to propagate only the subset $C_2 = \{s_1\}$ to upstream broker B_1.

In the experiments in Section V, we call RegularCover the above technique without top-k filtering employed.

B. Top-k model and ranking criteria

Consider a publication p that matches a set of subscriptions S according to content-based filtering semantics [25]. The problem addressed in this paper is to deliver the publication to the highest ranked k-sized subset of S as follows.

Relevance-based (scoring) semantics – The scoring function, $\mathrm{score}(p, s)$, computes a score value given a publication p and

Fig. 1. Subscription processing flow with covering

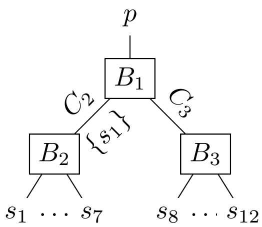


Fig. 2. Example with three brokers

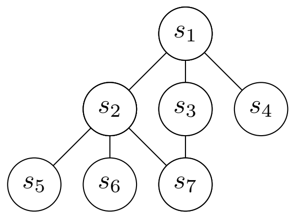


Fig. 3. Covering poset at B_2

i	x	y	area	rank
1	[1,10]	[1,10]	100	1
2	[1,10]	[1,5]	50	2
3	[3,7]	[1,8]	40	3
4	[9,10]	[9,10]	4	7
5	[1,5]	[1,5]	25	4
6	[3,10]	[2,4]	24	5
7	[3,6]	[1,5]	20	6

TABLE I
EXAMPLE SUBSCRIPTIONS AND SCORING AT B_2

subscription s. The subscriptions are ranked by descending values, such that the k subscriptions with the highest scores are selected. Table I shows the ranking of the example subscriptions using the x-y area as the score.

Figure 4 shows an overview of the processing flow at a broker using top-k filtering. Each incoming publication p is first matched against known subscriptions $\mathbb{S}$ under the usual pub/sub model (e.g., content-based predicates). The resulting set S of matching subscriptions is sorted according to the scoring semantics. This ranked set S of matching subscriptions is then filtered to retain only a k-sized subset of the highest ranked subscriptions. These subscriptions, along with the publication p, are provided to the usual routing component in order to forward p to the appropriate next hops towards the intended destinations (e.g., *downstream* brokers).

C. Semantics propagation

The top-k ranking semantics above are from the perspective of the publisher. In particular, the case where a subscriber chooses to receive only a subset of matching publications has been studied and not considered in this paper [10].

As a consequence of this decision, the publisher must specify a value k with each advertisement a it issues, indicating that publications induced by a should be sent to the k highest-ranked matching subscriptions. In addition, the scoring function $score(p, s)$ is issued either with the advertisement, or there could also be a system-wide ranking function. In this paper, we assume the most general case of a per-advertisement scoring function. Note that each individual publication can request a specific k value, as long as it is lower than the k defined by a.

For a publication p and set of matching subscriptions S, the scoring function $score(p, s)$ is called with each subscription $s \in S$. Based on the score, a rank in the range $[1, |S|]$ is assigned to each subscription.

We note that our distributed approach is compatible with any ranking function, such as those described in [22].

D. Naive solution

This paper focuses on the forwarding of advertisement, subscription, and publication messages to achieve top-k dissemination. Notably, the algorithm to compute the top-k set for a publication is assumed to be known. More specifically, given a set of subscriptions $\mathbb{S}$ and a publication p: (i) a matching algorithm can compute the set of subscriptions $S_p \subseteq \mathbb{S}$ that match p, and (ii) a top-k algorithm can use a provided $score(p, s)$ function to compute the k-sized set of top ranked subscriptions in S_p.

If there is no subscription covering, then each broker can compute the top-k ranked subscriptions locally and forward the publications accordingly along with the IDs of the subscriptions to deliver the publication to. We refer to this algorithm, when used without covering, as RegularK in Section V.

E. Problem with naive solution

The naive solution works because each broker knows the complete set of subscriptions in the system, allowing for the ranks of matching subscriptions to be computed. However, routing with covering forwards only a subset of subscriptions to upstream brokers. The subset is chosen to be an appropriate summary of the subscriptions such that the upstream broker can correctly determine which downstream brokers may have matching subscriptions. In other words, the subscription covering algorithm avoids forwarding subscriptions whose interest space is subsumed by another subscription [17].

Given a set of subscriptions, and their pairwise covering relationships, the broker constructs a poset that represents the covering relationships among its subscriptions. It then forwards only those subscriptions that are not covered by any other.

Consider once again the three brokers example in Figure 2, with the covering poset in Figure 3 for broker B_2. Suppose $k = 2$ is employed. When subscription covering is not employed, broker B_1 is aware of all the subscriptions and can select the appropriate top-k set as outlined in Section III-D. However, with subscription covering, broker B_2 only forwards the roots of the poset, subscription s_1 in this case, to the upstream broker B_1. Since broker B_1 does not have knowledge of all the subscriptions, it cannot accurately rank all the subscriptions (using their area) and therefore determine the k highest ranked subscriptions from B_2 and B_3. In Section IV, we resolve the problem by modifying the covering component of subscription forwarding (see the dotted box of Figure 1) which then allows publication processing to achieve correct top-k filtering while retaining most of the benefits of subscription covering.

IV. TOP-K FORWARDING ALGORITHMS

This section presents first a distributed top-k algorithm (RankCoverK) that correctly supports covering for relevance-based ranking, as opposed to the RegularK solution shown in

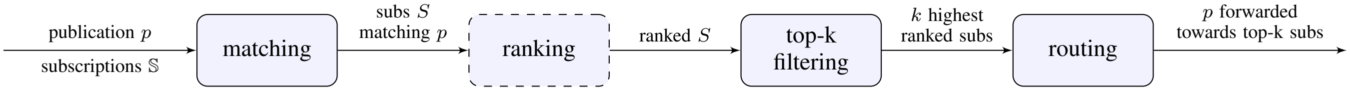


Fig. 4. Publication processing flow with top-k filtering

Section III-D. We also present AncestorCoverK, an optimized version of RankCoverK using ancestor counting. We provide pseudocode and proofs in our tech report[1].

We now describe top-k algorithms that support covering. The challenge is to decide how many of the k highest ranked subscriptions are reachable from each downstream broker.

Without loss of generality, we consider again the topology in Figure 2. Brokers B_2 and B_3 should employ covering and forward a subset of their subscriptions to B_1 sufficient for supporting top-k filtering. Suppose $k = 2$ for the publisher. Then, when a publication arrives at broker B_1, it needs to distinguish among three cases:

1) **2-0** case: The two highest ranked subscriptions are both reachable through broker B_2.
2) **1-1** case: One of the highest ranked subscriptions is reachable through broker B_2, and the other through B_3.
3) **0-2** case: The two highest ranked subscriptions are both reachable through broker B_3.

The key insight is that broker B_1 only needs to distinguish among the above three cases. In particular, it does not need to know if there are more than $k = 2$ matching subscriptions downstream of a particular broker. Therefore, it suffices for each broker to forward its own k highest ranking subscriptions for all possible publications that may be induced by advertisement a. Note that unlike the RegularK solution, each broker along the dissemination path must perform top-k filtering to ensure that covered subscriptions can be selected.

We will now generalize and formalize this point. Let S_j be the set of subscriptions that match publication p_j at a given broker, and let $s_{j,1}, ..., s_{j,n}$ be the subscriptions in S_j ordered by ranking. Furthermore, let $\hat{S}_j$ be the top-k set in S_j, that is, $\hat{S}_j = s_{j,1}, ..., s_{j,q}$ where $q = min(n, k)$. Finally, let $\hat{S} = \bigcup_j \hat{S}_j$ be the set of highest ranking subscriptions for all possible publications p_j.

We now outline several algorithms to compute $\hat{S}$, the set of subscriptions each broker must forward to its upstream broker in order to allow top-k filtering with subscription covering.

Rank-cover – Let subscriptions $s_1, \cdots, s_n$ be the subscriptions at a given broker that intersect an advertisement a from an upstream broker.

In this algorithm, the subscription cover is a poset built according to an *extended covering definition* that is based on the subscription predicates and given ranking functions. We call this the **rank-cover** definition. In particular, subscription s_i rank-covers s_j if and only if both these conditions are true:

(i) the predicates of s_i cover those of s_j. (This is the conventional covering definition.)

(ii) the rank of s_i covers that of s_j for advertisement a. More precisely, $\forall p_b, \{s_i, s_j\} \subseteq S_b \implies (ranking(s_i, S_b) < ranking(s_j, S_b))$, where p_b is any publication induced by advertisement a, and $ranking(s, S_b)$ returns the rank of a subscription s in the matching subscription set S_b for publication p_b.

Intuitively, rank-covering implies that if a publication matches s_j, then it will also match s_i with a higher rank.

Each broker then forwards all the subscriptions in the first k levels of the poset, where k is the desired number of top-k subscriptions. This will ensure that the upstream broker has enough subscriptions to determine how many of the top-k ranking subscriptions are downstream.

More precisely, this algorithm constructs $\hat{S}$ as $\{s_i | \text{depth}(s_i) \leq k\}$ where depth(s_i) is the depth of subscription s_i in the rank-cover poset.

For example, consider the topology in Figure 2, and let subscriptions $s_1, ..., s_7$ be the subscriptions at broker B_2 that intersect with advertisement a (with $k = 2$) from broker B_1. Figure 3 depicts the rank-covering poset for the subscriptions at B_2 that intersect advertisement a using area for scoring. Note that in this example, it is identical to the regular covering poset. Since $k = 2$, all nodes with depth ≤ 2 will be forwarded to B_1. So, subs s_1, s_2, s_3, s_4 will be sent to B_1. Broker B_1 now has enough subscriptions from B_2 to determine if 0, 1 or 2 of the top ranked subscriptions are from broker B_2.

The rank-covering algorithm is referred to as RankCoverK for the remainder of this paper.

Ancestor counting – The previous rank-cover covering algorithm can be optimized so that certain subscriptions, which have multiple parents, are removed from the forwarding set. This happens if there are enough overlapping ancestors to cover the subscription and satisfy k matches.

In this optimization, rather than forwarding all subscriptions in the poset with depth $\leq k$, we only forward those subscriptions with $< k$ ancestors in the poset. This algorithm constructs $\hat{S}$ as $\{s_i | \text{ancestors}(s_i) < k\}$ where ancestors(s_i) is the number of subscriptions in the rank-cover poset that are in the path from the root to s_i.

If $k = 3$ in the example from Figure 3, every subscription would be forwarded except s_7 which has 3 ancestors. This algorithm is referred to as AncestorCoverK.

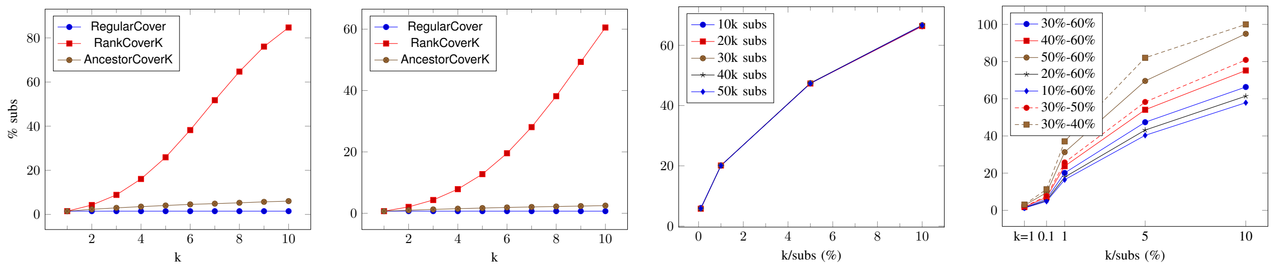
V. Evaluation

This section contains the results of experiments performed on our various algorithm implementations. We describe scalability experiments which compare the covering performance of three implementations: (1) our baseline top-k solution with covering (RankCoverK), (2) an optimized version using ancestor counting (AncestorCoverK), and (3) our baseline covering algorithm without top-k (RegularCover).

Setup – The implementation is performed in Java using the PADRES pub/sub prototype[2]. Experiments are performed on

[1]http://msrg.org/papers/topktr15

[2]http://padres.msrg.toronto.edu/



(a) 10000 subscribers (low k) (b) 50000 subscribers (low k) (c) AncestorCoverK (high k) (d) AncestorCoverK (10k subs)

Fig. 5. Percentage of forwarded subscriptions (varying k and subscription size)

the SciNet testbed using the General Purpose Cluster (GPC)[3]. Up to 60 distinct machines in a LAN are used, each equipped with Intel Xeon quad-core processors and 16GB RAM.

The publication dataset used is synthetic, with one publisher emitting 30 publications per minute. We employ a single publisher only since our measurements are unaffected by the number of publishers.

The broker overlay topology consists of 10 core brokers connected in a chain. Each core broker is then connected to 5 edge brokers. For our scalability experiments, we connect 200 to 1000 subscribers to each edge broker. We employ between 10000 to 50000 subscribers, each with one subscription. This setup is modeled as a network of data centers connected through gateways (core brokers) and measures the impact of our algorithms on delivery paths with multiple broker hops.

Subscriptions are randomly and uniformly generated to span between 30% and 60% of the publication space (subscription size). We argue that the large size of the subscriptions is appropriate for our motivating scenarios where users are receiving an overwhelming amount of data, hence, the need for additional top-k filtering. Furthermore, the narrow size range (30%-60%) creates conservative covering results with shallow and broad covering posets, since neither excessively large subscriptions exist to cover large spaces, nor are small subscriptions present that can easily be covered. We also conduct a sensitivity analysis where we vary the subscribe size range by varying the minimum (10%-60% to 50%-60%) or the maximum (30%-40% to 30%-60%).

We conduct the experiments using two sets of k. First, advertisements are evaluated using low values of k between 1 and 10. These low values are used for micro-benchmarking purposes. Second, advertisements are set with higher values of k ranging between 0.1% to 10% of the total number of subscriptions, which we refer to as the *selectivity*. For instance, in an experiment with 50000 subscriptions, a k set to 5000 has a selectivity of 10%. As top-k is useful for suppressing an overwhelming amount of data, increasing the k value beyond 10% would be semantically ineffective.

We employ a covering-agnostic random ranking function: each subscription known by the broker for a publication is assigned a uniformly random rank. In practice, our solutions are compatible with any scoring scheme.

Metric – To evaluate covering performance, we measure the reduction in the number of subscriptions sent upstream (i.e., subscription traffic towards publishers). We measure the ratio between the number of subscriptions at the publisher edge broker and the total number of subscriptions sent by all subscribers. Since the publisher edge broker is the last hop for every subscription in the system (i.e., every subscription is forwarded to the publisher edge broker if covering is not employed), this number is an overall measure for covering in the system. The lower bound is provided by our baseline covering algorithm RegularCover which does not support top-k, while the upper bound is 100% for top-k without covering (RegularK). Our proposed solutions fall somewhere in between and allow us to assess the impact of top-k dissemination on covering. Covering performance is then defined as the percentage of subscriptions covered and thus not forwarded ($100\% - forwarded\%$).

Results – Figures 5(a) and 5(b) show the covering performance of our various algorithms relative to the baseline RegularCover with low k values (1-10) and different amount of subscriptions (with size between 30%-60%). A lower number indicates that the covering algorithm has been more effective in reducing the number of subscriptions disseminated.

We first note that the baseline RegularCover propagates only 1.47% of 10000 subscriptions, and 0.69% for 50000 subscribers. Note that the baseline performance is unaffected by the value of k, since it does not support top-k. RegularCover demonstrates scalability through the improvement in subscription reduction with increasing subscription loads.

The graphs show that our rank-cover solution, RankCoverK, is sensitive to increasing values of k: the number of subscriptions forwarded increases from 1.47% to 84.71% when k increases from 1 to 10 with 10000 subscriptions. The performance declines rapidly and is nullified (100% subscriptions forwarded) when k reaches 19. In the 50000 subscriptions case, the number of subscriptions forwarded by RankCoverK increases from 0.69% to 60.53% in the 1-10 range, which is an 28.5% improvement over the 10000 subs case. This indicates that the solution scales with the number of subscriptions in the low k range. However, when k is set to 27, the covering performance is still reduced to 0% of subscriptions covered. This indicates that RankCoverK can only operate at very low k values; there is a fixed limit for k which does not scale with the number of subscribers. This limitation limits the applicability of RankCoverK, as queries typically select a k as a function of the number of subscriptions (i.e., % selectivity). As described in the setup, we selected subscriptions such that the covering posets are shallow and broad. Therefore, traversing a few levels deep is enough to encounter a large number of subscriptions, which explains the significant impact of k on the covering performance.

[3]http://www.scinet.utoronto.ca/

Our optimized solution, AncestorCoverK, eliminates this limitation by demonstrating covering performance across a wide range of k. In the low range of 1-10, the covering performance stays stable at under 6.03% for 10000 subscriptions and under 2.5% for 50000 subscriptions. AncestorCoverK exhibits the same performance benefits as RankCoverK with respect to the number of subscriptions, while removing the dependency on low values of k. This optimization is well-suited for our workload, since it contains many subscriptions residing at a shallow depth, but covered by multiple parents.

In Figure 5(c), we further test the scalability of AncestorCoverK with regards to k with values between 0.1% to 10%. For example, the experiment with 50000 subscriptions used a k between 50 to 5000. The results show that the covering performance decreases from over 94.24% to 33.3% when in that k range. This indicates that our optimized solution is still sensitive to higher values of k, albeit to a far lesser degree than RankCoverK. The performance is also stable with regards to increased number of subscriptions, with differences of less than 0.4% across different runs.

In Figure 5(d), we vary the subscription size from a larger range (10%-60%) to a smaller range (30%-40% or 50%-60%) using AncestorCoverK at 10000 subscriptions. We first note that the subscription size also affects the baseline RegularCover: the performance varies between 1.22% for the largest range used (10%-60%) to 3.08% for the narrowest range used (30%-40%). When the subscriptions are similarly sized, the covering optimization decreases since it requires large subscriptions to be able to cover smaller ones. For AncestorCoverK, the same pattern can be observed. For instance, 40.33% of subscriptions are forwarded for the range 10%-60% at 5% selectivity, compared to 82.05% for 30%-40% at 5% selectivity. Also note that while 30%-40% and 50%-60% both exhibit a range of 10%, 50%-60% performs better (69.59% at 5% selectivity). This is because having larger subscriptions in general increases the chance of covering occurring, since subscriptions are more likely to overlap. We note there is 2.52 times increase in number of subscriptions forwarded for RegularCover from 10%-60% to 30%-40%, whereas there is only a 2.03 times increase for AncestorCoverK. We can therefore conclude that AncestorCoverK is not more sensitive to subscription size than the baseline RegularCover which does not support top-k.

Summary: Both solutions are sensitive to varying degrees to the value of the parameter k. In particular, RankCoverK is bounded by a k of 27, after which no covering is obtained regardless of the number of subscriptions. This is due to the broad and shallow nature of the covering tree, where most subscriptions can be found by traversing at a low k depth. On the other hand, AncestorCoverK retains 81% of the covering benefits of RegularCover at a k set to 1% selectivity, which is 5000 for 50000 subscribers. This range of k is practical for a variety of studied workloads, which typically have 1-2% selectivity [26]. Furthermore, both solutions are scalable to the number of subscriptions, as the covering performance stays stable in that regards. Finally, AncestorCoverK is at least as sensitive than the baseline RegularCover when varying the range of subscription sizes.

VI. Conclusions

Many applications found in domains such as targeted marketing require a communication middleware that supports selective filtering of data. This paper extends pub/sub to support relevance-based top-k semantics along with subscription covering. We introduce RankCoverK, a novel rank-cover algorithm to construct an efficient covering partially ordered set (poset) which supports top-k relevance scoring. We further develop AncestorCoverK, an ancestor counting optimization to reduce the cover size. We conclude with a scalability analysis of our algorithms, incorporated in an open-source distributed publish/subscribe system, that demonstrates the performance of our solutions. AncestorCoverK is scalable and retains over 81% of the covering benefit when using a k set at 1% selectivity.

VII. Acknowledgments

This research was supported by the Alexander von Humboldt Foundation.

References

[1] "The IBM strategy," http://www.ibm.com/annualreport/2013/, 2013.
[2] T. Yan and H. Garcia-molina, "Index structures for selective dissemination of information under the boolean model," *TODS'94*.
[3] M. K. Aguilera, R. E. Strom, D. C. Sturman, M. Astley, and T. D. Chandra, "Matching events in a content-based subscription system," in *PODC'99*.
[4] F. Fabret, H.-A. Jacobsen, F. Llirbat, J. Pereira, K. A. Ross, and D. Shasha, "Filtering algorithms and implementation for fast pub/sub systems," in *SIGMOD'01*.
[5] A. Campailla, S. Chaki, E. Clarke, S. Jha, and H. Veith, "Efficient filtering in publish-subscribe systems using binary decision diagrams," in *ICSE'01*.
[6] S. Whang, C. Brower, J. Shanmugasundaram, S. Vassilvitskii, E. Vee, R. Yerneni, and H. Garcia-Molina, "Indexing boolean expressions," in *VLDB'09*.
[7] M. Fontoura, S. Sadanandan, J. Shanmugasundaram, S. Vassilvitski, E. Vee, S. Venkatesan, and J. Zien, "Efficiently evaluating complex Boolean expressions," in *SIGMOD'10*.
[8] M. Sadoghi and H.-A. Jacobsen, "BE-Tree: An index structure to efficiently match Boolean expressions over high-dimensional discrete space," in *SIGMOD'11*.
[9] ——, "Relevance matters: Capitalizing on less (top-k matching in publish/subscribe)," in *ICDE'12*.
[10] K. Zhang, M. Sadoghi, V. Muthusamy, and H. Jacobsen, "Distributed ranked data dissemination in social networks," in *ICDCS'13*.
[11] M. Sadoghi and H. Jacobsen, "Analysis and optimization for Boolean expression indexing," *ACM Trans. Database Syst.'13*.
[12] W. Culhane, K. R. Jayaram, and P. Eugster, "Fast, expressive top-k matching," in *Middleware '14*.
[13] L. Chen, G. Cong, X. Cao, and K. L. Tan, "Temporal spatial-keyword top-k publish/subscribe," in *ICDE'15*.
[14] X. Wang, Y. Zhang, W. Zhang, X. Lin, and Z. Huang, "Skype: Top-k spatial-keyword publish/subscribe over sliding window," in *VLDB'16*.
[15] V. Setty, G. Kreitz, R. Vitenberg, M. van Steen, G. Urdaneta, and S. Gimåker, "The hidden pub/sub of spotify: (industry article)," in *DEBS'13*.
[16] X. Bai, R. Guerraoui, and A. Kermarrec, "Personalizing top-k processing online in a peer-to-peer social tagging network," *TOIT'14*.
[17] G. Li, S. Huo, and H.-A. Jacobsen, "A Unified Approach to Routing, Covering and Merging in Publish/Subscribe Systems Based on Modified Binary Decision Diagrams," in *ICDCS'05*.
[18] R. Fagin, A. Lotem, and M. Naor, "Optimal aggregation algorithms for middleware," in *PODS'01*.
[19] I. F. Ilyas, G. Beskales, and M. A. Soliman, "A survey of top-k query processing techniques in relational database systems," *Comput. Surv.'08*.
[20] K. Pripužić, I. P. Žarko, and K. Aberer, "Top-k/w publish/subscribe: finding k most relevant publications in sliding time window w," in *DEBS'08*.
[21] M. Drosou, E. Pitoura, and K. Stefanidis, "Preferential publish/subscribe," in *PersDB'08*.
[22] M. Drosou, K. Stefanidis, and E. Pitoura, "Preference-aware publish/subscribe delivery with diversity," in *DEBS'09*.
[23] A. Machanavajjhala, E. Vee, M. Garofalakis, and J. Shanmugasundaram, "Scalable ranked publish/subscribe," in *VLDB'08*.
[24] L. Chen and G. Cong, "Diversity-aware top-k publish/subscribe for text stream," in *SIGMOD '15*.
[25] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, "Design and evaluation of a wide-area event notification service," *TOCS'01*.
[26] N. K. Pandey, K. Zhang, S. Weiss, H.-A. Jacobsen, and R. Vitenberg, "Minimizing the Communication Cost of Aggregation in Publish/Subscribe Systems," in *ICDCS'15*.