Dimension Transform Based Efficient Event Filtering for Symmetric Publish /Subscribe System

Botao Wang[1] and Masaru Kitsuregawa[1]

Institute of Industrial Science, The University of Tokyo
Komaba 4–6–1, Meguro Ku, Tokyo, 135–8505 Japan
{botaow, kitsure}@tkl.iis.u-tokyo.ac.jp

Abstract. There exists a class of publish/subscribe applications, such as recruitment, insurance, personal service, classified advertisement, electronic commerce, etc., where publisher needs the capability to select subscribers. Such kinds of publish/subscribe applications are called symmetric publish/subscribe system. The existing event matching algorithms designed for traditional publish/ subscribe systems (called asymmetric publish/subscribe system) can not be applied to symmetric publish/subscribe systems efficiently.
By extending the existing data model and algorithm, we propose an event matching method for symmetric publish/subscribe system based on dimension transform regarding the query in multidimensional space. An efficient underlying multidimensional index structure is chosen and verified. Our proposal is evaluated in a simulated environment. The results show that, our proposal outperforms the other possible solutions in one or two orders of magnitude. For a typical workload containing one million subscriptions with 16 attributes, an event can be filtered within several milliseconds and the subscription base can be updated within hundreds of microseconds. We can say that our proposal is efficient and practical for symmetric publish/subscribe systems.

1 Introduction

There exists a class of publish/subscribe applications, such as recruitment, insurance, personal service, classified advertisement, electronic commerce, etc., where publisher needs the capability to select subscribers who can receive its publications. Consider the recruitment for example, company is publisher and job seeker is subscriber. For a complete recruitment matching, besides providing with working conditions, the publisher (company) wants to check the information related to the subscriber (job seeker) also. For example, the subscriber is required to be older than 18. Accordingly, the subscriber needs to provide his own information also. Such kinds of systems are called *symmetric* publish/subscribe system. Different from traditional publish/subscribe system (called *asymmetric* publish/subscribe system in this paper), both subscriber and publisher keep information and filtering criteria as shown below.
$(S1)$ $Subscription:$
$Criteria$:$(Salary > 500)$ AND $(Location = Paris)$, $Information$:$(Age, 25)$
$(E1)$ $Event:$

$Information$:($Salary, 400 - 600$), ($Location, Paris$), $Criteria$: ($Age > 30$)

In the context of event matching, many event matching techniques [4] [5] [7] [8] [13] [16] [17] have been proposed. The main challenge here is that all above techniques are designed for asymmetric publish/subscribe system. These techniques can not be applied to symmetric publish/subscribe system for three reasons: 1) Predicates are defined and included in event. 2) Not only point data (*Location* in $E1$) represented by one constant but also range data ($Salary$ in $E1$) represented by a pair of constants need to be supported. 3) The frequency of index updating is same as that of event arriving. The new type of event matching techniques for symmetric publish/subscribe system is required.

As far as we know, the symmetric publish/subscribe system is first introduced in [12] without performance evaluation. In this paper, for symmetric publish/subscribe system, we propose and evaluate an efficient event matching method based on a multidimensional index structure MultiLevel Grid File (MLGF) [14] [15] with dimension transform.

The main contributions of this paper are that: 1) Propose an event matching model for symmetric publish/subscribe system, which allows information and criteria to be defined in both event and subscription. The format of information is extended from point format to range format. 2) Extend the dimension transform techniques used in [13] to support event matching for both symmetric and asymmetric publish/subscribe systems. Moreover, different from [13], where UBTree [2] [3] [10] [11] is chosen as its underlying index structure, we propose a more efficient multidimensional index structure MLGF, the performance of event matching can be improved one order of magnitude in almost all cases.

The remainder of this paper is organized as follows. Section 2 defines the event matching model of symmetric publish/subscribe system. Section 3 introduces the main idea of our proposal after analyzing the limitations of the related solutions. Section 4 describes the method of the dimension transform to support event matching of symmetric publish/subscribe system. Section 5 introduces the related work. Section 6 reports experimental evaluation. Finally, conclusions are presented in Section 7.

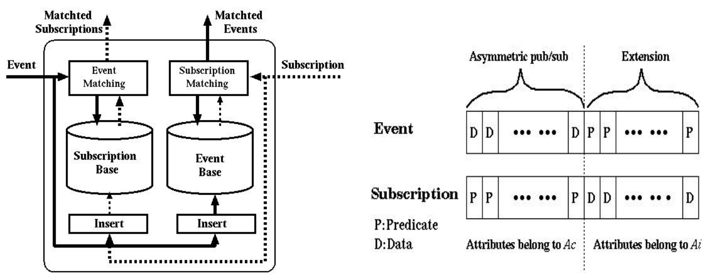
2 Event Matching of Symmetric Publish/Subscribe System

2.1 Architecture of Symmetric Publish/Subscribe System

The architecture of symmetric publish/subscribe system is shown in Fig.1-a. There event and subscription are "symmetric", and the roles of subscriber and publisher are relative not absolute. While an event arrives, besides matching the event on subscription base, the system inserts the event to event base also. The frequency of event (subscription) matching operations is same as that of data updating of event (subscription) base. Symmetric publish/subscribe system should support both high rate of event matching and high rate of data updating.

2.2 Data Model of Symmetric Publish/ Subscribe System

Schema Attributes The schema attributes are defined from the view of subscriber. Let Ac, Ai and A denote the filtering criteria domain, the information



(a) Architecture of symmetric publish/subscribe

(b) Data structure of event and subscription

Fig. 1. The symmetric publish/subscribe system

domain and the publish/subscribe application domain, the schema attribtues are defined as following:

$Ac = \{ac_1, ac_2, ..., ac_g, ..., ac_i\}$ $\quad 1 <= i,\ ac_g \in Ac,\ 1 <= g <= i$
$Ai = \{ai_1, ai_2, .., ai_h, ..., ai_j\}$ $\quad 0 <= j,\ ai_h \in Ai,\ 0 <= h <= j$
$A = Ac \cup Ai$

Event and Subscription Both event and subscription in symmetric publish/subscribe system consist of filtering criteria and information data as shown in Fig.1-b. Because range data (interval) must be supported and predicate can be represented as an interval also, an event e and a subscription s are defined as conjunctions of intervals in the following formats:

e=$\{(a_1 : EI_{a_1}), ..., (a_k : EI_{a_k}), ..., (a_{i+j} : EI_{a_{i+j}})\}$
s=$\{(a_1 : SI_{a_1}), ..., (a_k : SI_{a_k}), ..., (a_{i+j} : SI_{a_{i+j}})\}$

where $a_k \in A = Ac \cup Ai, 1 <= k <= i + j$, and EI_{a_k}, SI_{a_k} are the intervals in application domain respectively. The interval has format:

$Ia = [Is_a, Ie_a]$

where $a \in A$, $Is_a, Ie_a \in a$, $Is_a <= Ie_a$. Notice that both predicate and information data are represented by the same interval format here. They are different semantically.

Event Matching An event pair (a, EI_a) matches a subscription pair (a, SI_a) if EI_a intersects SI_a. An event e satisfies a subscription s if all subscription pairs in s are matched by its corresponding event pairs.

3 Solution Overview

3.1 Limitations of Related Solutions

Compound Algorithm As far as we know, the symmetric publish/subscribe system was first introduced in [12] without performance evaluation. The events are matched based on two indexes: predicate index (Count algorithm [16] or

Handson [7] [8]) built on data belong to *Ac* and data index (B+tree) built on data belong to *Ai*. The event data is divided into two subsets (Fig.1-b) : the data subset belong to *Ac* and the data subset belong to *Ai*. The two data subsets are sent to their corresponding indexes and the result subscriptions are obtained by joining two intermediate result sets from two indexes.

The main problem of the Compound algorithm is performance. Both predicate index and data index are a cluster of one-dimensional structures there. As analyized and evaluated in [3] [10] [13] [17], the performance of such kind of clusters is very sensitive to the selectivity of attributes. According to the analyses and evaluation results in [3] [10], it is hard to expect competitive performance with multiple B+trees compared to the multidimensional index structures like UBTree. As introduced in [13] [17], the performances of UBTree-based and RTree-based event matching are three orders of magnitude faster than that of the Count algorithm [16] in most of cases regarding different workloads. Logically, the Count algorithm [16] and the Hanson algorithm [7] [8] have same complexity order for event matching.

Algorithms Based on Multidimensional Indexes Multidimensional indexes, like UBTree [3] [10] and RTree [6] are feasible for event matching of asymmetric publish/subscribe system as introduced in [13] [17]. The dimension transform is adopted in order to avoid overlap in hypercubes corresponding to subscriptions. There the hypercubes in d space are transferred into points in $2d$ space. The point access method used there is UBTree.

The problems of [13] [17] are that: 1) The method introduced in [13] can not be applied to symmetric publish/subscribe directly. The reason is that the new data type (range data) and operation (predicate) in event are newly defined in symmetric publish/subscribe system, and event matching here is an intersection query based on the model (Section 2.2) instead of a point enclosed query [13] which corresponds to asymmetric publish/subscribe system . 2) For the range search based on UBTree which was chosen as the underlying index structure in [13], the number of empty spaces (no data is kept there) becomes larger with increasement of dimension number. Skipping empty space is an expensive memory operation, which can not be neglected in the event matching based on main memory structure.

3.2 Main Idea

The main idea of our solution is stated as: the event matching of symmetric publish/subscribe system is regraded as an intersection query on hypercubes in d space and the intersection query on hypercubes is transformed into a range query on points in $2d$ space so as to make use of efficient point access methods for event matching.

Different from [13] [17], the dimension transform (to be introduced in Section 4) supports event matching of symmetric publish/subscribe system. Moreover, instead of UBTree, we propose to use a more efficient underlying index structure, MultiLevel Grid File (MLGF) [14] [15]. MLGF is a dynamic, balanced, multidimensional index structure that adapts to nonuniform and correlated distributions. For the details of Multilevel Grid File, please refer to [14] [15].

4 Dimension Transform

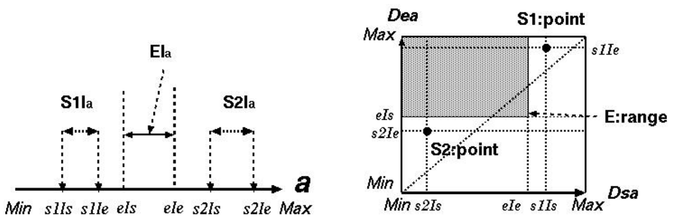


Fig. 2. Dimension Transform

4.1 Dimension Transform for Symmetric Publish/Subscribe System

As defined in Section 2.2, both subscription and event are conjunctions of intervals. The method of dimension transform is same for all attributes. In the follows, we introduce the dimension transform method for one attribute. The same intervals

$EI_a = [EIs_a, EIe_a], SI_a = [SIs_a, SIe_a]$

defined in Section 2.2 is used here. Given an attribute $a \in A$ with domain size $[Min, Max]$, two new dimensions in a 2d space Dsa and Dea are defined corresponding to starting and ending points.

We start from the cases that EI_a doesn't intersect with SI_a, which means the predicate is not matched. As shown in Fig.2, there are only two cases which are represented by two pairs of intervals $(EI_a, S1I_a)$ and $(EI_a, S2I_a)$. Logically, the two cases can be summarized as follows,

$EIs_a > SIe_a$ OR $EIe_a < SIs_a$

From above expression, we can deduce the expression representing intersection of two intervals:

$$\text{NOT}\ (EIs_a > SIe_a \text{ OR } EIe_a < SIs_a)$$
$$\Updownarrow$$
$$(EIs_a <= SIe_a \text{ AND } EIe_a >= SIs_a)$$
$$\Updownarrow$$
$$(EIs_a <= SIe_a <= Max) \text{ AND } (Min <= SIs_a <= EIe_a)$$

If SIs_a and SIe_a are considered as one point in the 2d space, by mapping SIs_a and SIe_a to the newly defined two dimensions Dsa and Dea, one 1d intersection query on hypercubes can be transformed into one 2d range query on points as shown in Fig.2. The range in 2d space is

$Range : ([Min, EIe_a], [EIs_a, Max])$

and the point is

$Point : (SIs_a, SIe_a)$

5 Related Work

A lot of algorithms related to event matching have been proposed. They are proposed for publish/subscribe systems [1] [5] [9] [13] [16] [17], continuous queries [4] and active databases [7] [8].

Predicate indexing techniques have been widely applied. There, a set of one-dimensional index structures are used to index the predicates in subscriptions. Mainly, there are two kinds of algorithms based on multiple one-dimensional index structures: Count algorithm [16] and Hanson algorithm [7] [8]. The performances of Count algorithm and Hanson algorithm have same complexity order, they differ from each other by whether or not all predicates in subscriptions are placed in the index structures. Meanwhile in [13] [17], the event matching based on multidimensional index structures has been proved to be feasible and efficient compared to the Count algorithm. The conclusions of [13] [17] are the basis of this paper. Hanson algorithm is extended in [5] where subscriptions were clustered according to their equality predicates and mutli-attribute hashing was utilized to find the related clusters. The size of domain can not be too larger for reason of multi-attribute hashing.

The testing networking based techniques [1] [9] initially preprocess the subscriptions into a matching tree. Different from the predicate indexes, [1] and [9] built subscription index trees based on subscription schema. They suffer from the problems of space and maintenance.

Event matching is one critical step of continuous queries. In [4], predicate index was built based on Red-Black tree, there algorithm is similar to bruteforce which scans the total Red-Black tree every time when an event arrives.

All above algorithms are designed for asymmetric event matching, which can not be applied to symmetric publish/subscribe system directly.

6 Evaluation

6.1 Envaluation Environment

Parameter	**Value range**	**Default value**
Global parameters		
Number of subscriptions	0-2,500,000	1,000,000
Number of attributes (dimension)	8-64	16
Ratio of attributes belong to Ac to attributes belong to Ai	16:0-0:16	8:8
Ratio of one subscription is matched (selectivity)	0.001-0.05%	0.01%
Parameters related to subscription or event		
Ratio of an attributes to be used to define predicate	0-100%	100%
Ratio of equality predicates to be defined	0-100%	50%
Ratio of point data to be defined	0-100%	50%

Table 1. Simulated parameters

Three kinds of solutions have been implemented and compared based on main memory structure: 1) Naive. RTree is used directly for intersection query without dimension transform. 2) Compound. It is an implementation similar to the algorithm proposed in [12]. Different from the original proposal, two indexes for predicates and information data kept in subscriptions are two RTrees. 3) Dimension Transform. Two point access methods are chosen: UBTree and MLGF.

The events and subscriptions are created according to a workload specification. The parameters used in the evaluations are tabulated in Table.1 along with their range and default values. For each test, we change one parameter and fix the others with their default values without specific introduction. For each test, the average response time of 1000 inputs is measured.

All the solutions are implemented in C++. The type of all attributes is short integer. The fanout of UBTree, MLGF tree and R-tree[1] are 200, 20, 10. With these values, the best response times are obtained in one preliminary test with a workload of 1 million subscriptions and 10 thousands events in a 16d space. The hardware platform is a Sun Fire 4800 workstation with four 900MHz CPUs and 16G bytes memory under Solaris 8.

6.2 Evaluation Results

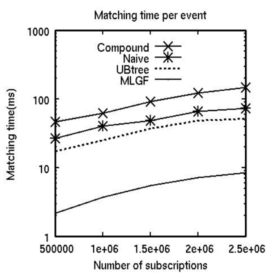
Performances Related to Event Matching Fig.3-a shows the scalability on the number of subscriptions. All solutions have good scalabilities here.

Fig.3-b shows the performances with different numbers of dimensions. Compared to the performance of *Compound*, the performance of *Naive* deteriorates quickly than that of *Compound*. The reason is that the number of dimensions used in *Naive* is double of that used in *Compound* (two RTrees are used). The influence of number of dimensions on RTree's performance accelerates when the dimension number becomes larger. The total number of intermediate results obtained from two indexes of *Compound* has an average value 2789 when dimension number is 8 and an average value 205 when dimension number is 16. That is the reason why its performance seems to upgrade a little when dimension number is 16, because the number of the intermediate results decreases 10 times here. Contrast to our expectation, the performance of MLGF does not changes linearly with number of dimension. The main reason is that the selectivity is fixed by default and the number of candidate objects which were filtered to get final results, decreases with the number of dimensions here.

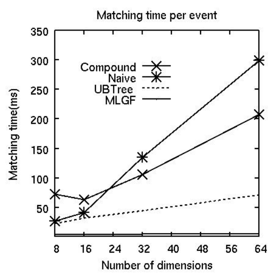
Fig.3-c shows the performance with different selectivities. Here 1 million subscriptions are used and the number of results for one event matching changes from 10 to 500. Except *Naive*, the time costs of other solutions become larger with the increasement of the selectivity. The reason that *Naive* is relatively stable is the numbers of candidate objects are on same order of amount.

Fig.3-d shows the influence of attributes distribution related to the numbers of attributes defined in *Ac* and *Ai*. "0:16" means the size of *Ac* is 0 and the size of *Ai* is 16. Subscriptions consist of information data only and events consist of predicates only. In this case, the event matching of symmetric publish/subscribe is same as a traditional query which is applied on static data (subscriptions). "16:0" means the size of *Ac* is 16 and the size of *Ai* is 0. It means that subscriptions consist of predicates only and event consists of information data only, which is similar to asymmetric publish/subscribe system. The difference is that the information data kept in events include has format of range data (Section

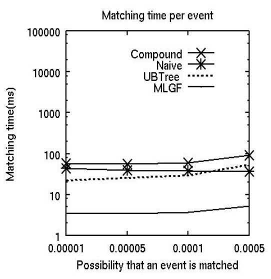
[1] Version 0.62b. http: //www.cs.ucr.edu/ marioh/ spatialindex. Only the two parameters related to fanout are changed here. The others are default values.



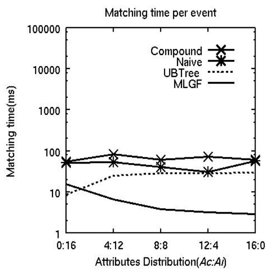
(a) Scalability on index size



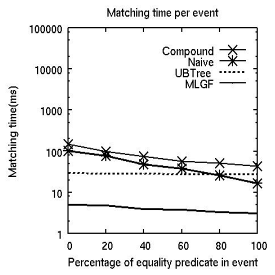
(b) Scalability on dimension



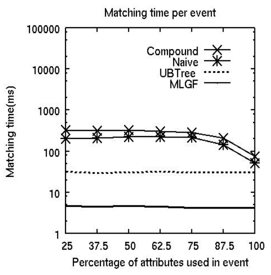
(c) Performance with different selectivities



(d) Performance with different distribution ratios of attributes



(e) Performance with different distribution ratios of event predicate



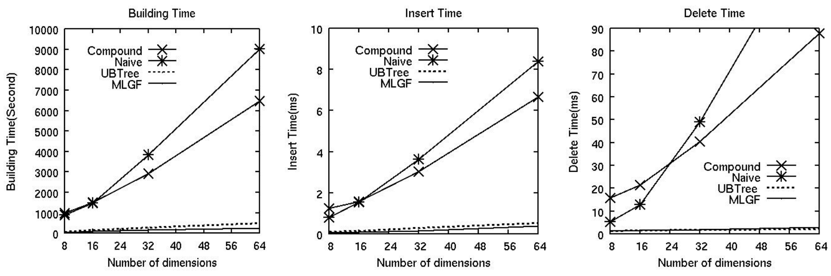
(f) Performance with different numbers of attributes used in event predicates

Fig. 3. Performances related to event matching

2.2). In above two cases, *Compound* method creates only one index (predicate or data) same as *naive*, so the performances of them are same. The performance difference between MLGF-based solution and UBTree-based solution becomes larger with increasement of *Ac*'s size. It indicates that MLGF is more suitable than UBTree for both symmetric and asymmetric publish/subscribe system.

Fig.3-e shows that the distribution ratio of different predicates (equality or non-equality) used in the events influences the performance bitterly. Except *UBTree*-based solution, the performances of other solutions become better while the percentage of equality predicates becomes larger. It is a process that the corresponding query changes from intersection query (symmetric publish/subscribe) to point enclosed query (asymmetric publish/subscribe), because equality predicate can be represented in the format of point data.

Fig.3-f shows that the changing of the number of unused attributes in event predicate only influences the performance of *Compound* and *Naive*. The performances become better for the reason that the more the number of used attributes is , the less the overlap is. But it does not influence the performance of $UBTree$ and $MLGF$ where dimension transform has been done.



(a) Index building time with different dimensions

(b) Performance of insertion with different dimensions

(c) Performance of deletion with different dimensions

Fig. 4. Performances related to index updating

Performances Related to Index Updating As introduced in Section 2.1, symmetric publish/subscribe system must support high dynamically updating operations. Fig.4-a shows that the index building time based on $UBTree$ and $MLGF$ increases linearly with the number of dimensions. In contrast, $Compound$ and $Naive$ deteriorate quickly for the reason of heavy overlaps.

Fig.4-b and Fig.4-c show the performances of insert operations and delete operations respectively. Time costs of $Compound$ and $Naive$ increase exponentially with the number of dimensions because Rtree is built based on overlap of spatial objects. The insert operation of $MLGF$ is a little faster than that of $UBTree$ and the delete operation of $MLGF$ is a little slower than that of $UBTree$. The reason is their different partition strategies. Merging two regions of $MLGF$ is a little expensive than merging two nodes of $UBTree$. The time costs of $MLGF$ and $UBTree$ increase linearly with the number of dimensions.

7 Conclusions

In this paper, we have described the event matching problem of symmetric publish/subscribe system and discussed the strategies of applying multidimensional index structures to symmetric publish/subscribe system. The key feature of our solution is that map the intersection query on hypercubes to a range query on points with dimension transform so as the efficient point access method (Multi-Level Grid File) can be utilized for event matching.

Three kinds of solutions based on RTree, UBTree and MLGF, were evaluated and compared with various workloads in a simulated environment. The results show that our proposal outperforms the others in one or two orders of magnitude in almost all cases regarding different workloads. Performance studies show that an event can be filtered within several milliseconds and subscriptions can be updated within hundreds of microseconds for a typical workload containing one million subscriptions with 16 attributes. We can say that our proposal is efficient and practical for symmetric publish/subscribe applications with high rates of incoming events and high rates of data changes.

References

[1] M. K. Aguilera, R. E. Strom, D. C. Sturman, M. Astley, and T. D. Chandra. Matching events in a content-based subscription system. In *The 18th annual ACM symposium on Principles of distributed computing*, pages 53–61, 1999.

[2] R. Bayer. The universal b-tree for multidimensional indexing. Technical Report TUM-I9637, Technische Universitat Munchen, November 1996.

[3] R. Bayer and V. Markl. The ub-tree: Performance of multidimensional range queries. Technical Report TUM-I9814, Technische Universitat Munchen, June 1998.

[4] S. Chandrasekaran and M. J. Franklin:. Streaming queries over streaming data. In *VLDB*, pages 203–214, 2001.

[5] F. Fabret, H. A. Jacobsen, F. Llirbat, J. Pereira, K. A. Ross, and D. Shasha. Filtering algorithms and implementation for very fast publish/subscribe systems. In *SIGMOD*, pages 115–126, 2001.

[6] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *SIGMOD*, pages 47–57, 1984.

[7] E. N. Hanson, C. Carnes, L. Huang, M. Konyala, L. Noronha, S. Parthasarathy, J. B. Park, and A. Vernon. Scalable trigger processing. In *ICDE*, pages 266–275, 1999.

[8] E. N. Hanson, M. Chaabouni, C.-H. Kim, and Y.-W. Wang. A predicate matching algorithm for database rule systems. In *SIGMOD*, pages 271–280, 1990.

[9] A. Hinze and S. Bittner. Efficient distribution-based event filtering. In *Workshops: 1st International Workshop on Distributed Event-Based Systems(DEBS)*, IEEE Computer Socienty, 2002.

[10] V. Markl. *MISTRAL:Processing Relational Queries using a Multidimensional Access Tecnnique*. PhD thesis, Technische Universitat Munchen, 1999. Published by infix Verlag, St.Augustin. DISDBIS 59, ISBN 3-89601-459-5.

[11] F. Ramsak, V. Markl, R. Fenk, M. Zirkel, K. Elhardt, and R. Bayer. Integrating the ub-tree into a database system kernel. In *VLDB*, pages 263–272, 2000.

[12] W. Rjaibi, K. R. Dittrich, and D. Jaepel. Event matching in symmetric subscription systems. In *Proceedings of the 2002 conference of the Centre for Advanced Studies on Collaborative research*, page 9. IBM Press, 2002.

[13] B. Wang, W. Zhang, and M. Kitsuregawa. UB-Tree based efficient predicate index with dimension transform for pub/sub system. In *DASFAA*, pages 63–37, 2004.

[14] K. Y. Whang, S. W. Kim, and G. Wiederhold. Dynamic maintenance of data distribution for selectivity estimation. *The VLDB Journal*, 3(1):29–51, 1994.

[15] K.-Y. Whang and R. Krishnamurthy. The multilevel grid file - a dynamic hierarchical multidimensional file structure. In *Proceedings of the Second International Symposium on Database Systems for Advanced Applications*, pages 449–459, 1992.

[16] T. W. Yan and H. Garcia-Molina. The sift information dissemination system. *ACM Trans. Database Syst.*, 24(4):529–565, 1999.

[17] W. Zhang. Performance analysis of Ub-tree indexed publish/subscribe system. Master's thesis, Department of Information and Communication Engineering, The University of Tokyo, 2004.