

# Location-based Matching in Publish/Subscribe Revisited

Mohammad Sadoghi, Hans-Arno Jacobsen  
Middleware Systems Research Group  
Department of Computer Science  
University of Toronto, Canada

## ABSTRACT

Event processing is gaining rising interest in industry and in academia. The common application pattern is that event processing agents publish events while other agents subscribe to events of interest. Extensive research has been devoted to developing efficient and scalable algorithms to match events with subscribers' interests. The predominant abstraction used in this context is the content-based publish/subscribe (pub/sub) paradigm for modeling an event processing application. Applications that have been referenced in this space include emerging applications in co-spaces that rely on location-based information [1, 7], algorithmic trading and (financial) data dissemination [13], and intrusion detection system [4]. In this work, we focus primarily on the role of state-of-the-art matching algorithms in location-based pub/sub applications.

## Categories and Subject Descriptors

H.3.3 [Information Search and Retrieval]: Information filtering

## General Terms

Algorithms, Design, Measurement, Experimentation, Performance

## Keywords

(Location-based) Publish/Subscribe, Complex Event Processing

## 1. INTRODUCTION

The co-existence of virtual and physical worlds brings unique opportunities for a new generation of applications. Applications that use information such as location gathered from the virtual world to continuously enrich users' physical world experience while using the real-time information gathered from the physical world to refresh the virtual world in turn [1, 7]. Examples of co-space applications are location-driven marketplace applications that allow virtual and physical shoppers to compete (bid on the last item) or cooperate (buy one get one free), location-based gaming that changes the gamer's environment relative to the gamer's physical location, and social networking applications that detect when vir-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

*Middleware 2012 Posters and Demos Track* December 3-7, 2012, Montreal, Quebec, Canada.

Copyright 2012 ACM 978-1-4503-1612-5/12/12 ...\$15.00.

tual friends are within a close proximity and initiate a physical interaction among them [7].

The efficient processing of large collections of patterns (e.g., Boolean expressions) over data streams plays a central role in major location-based services and other data intensive applications ranging from user-centric processing and personalization to real-time data analysis. For instance, emerging user-centric applications, including interactions in co-spaces and location-aware computational advertising, demand determining and presenting to an end-user only the most relevant content with respect to users' physical location. We achieve these user-centric requirements through novel high-dimensional indexing structures.

Some of the key challenges in efficient Boolean expression indexing are (1) scaling to millions of Boolean expressions defined over a high-dimensional space (2) enabling dynamic insertion and deletion of expressions, and, (3) indexing expressions enhanced with location-based predicates.

Although there has been a large body of work on developing efficient matching algorithms (e.g., [2, 3, 9, 10, 5, 4, 6, 12, 8]), the problem of location-based matching has received little attention. In fact, there is a myth that existing techniques are not suitable for real-time location-based matching and new algorithms are needed [6]. Contrary, to general belief, in this paper, we demonstrate that an existing state-of-the-art matching algorithm, i.e., BE-Tree [10], seamlessly scales when applied to location-based settings.

The core contributions of this work is to formalize the location-based Boolean expression matching problem and to present experimental evidence that establishes the effectiveness of existing matching techniques for location-based settings.

## 2. RELATED WORK

Problems related to indexing Boolean expressions have been studied in many contexts such as publish/subscribe matching [2, 3, 10, 5, 6] and XPath/XML matching [9]. In general, two main categories of matching algorithms have been proposed: counting-based [3, 5, 6] and tree-based [2, 10] approaches. These approaches can further be classified as key-based, in which for each expression a set of predicates are chosen as identifier [3]. Counting-based methods aim to minimize the number of predicate evaluations by constructing an inverted index over all unique predicates. One of the most efficient counting-based algorithms are Propagation [3], a key-based method. Likewise, tree-based methods are designed to reduce predicate evaluations and to recursively divide the search space by eliminating subscriptions on encountering unsatisfiable predicates. The most prominent tree-based method, BE-Tree, is a dynamic key-based algorithm [10], which is shown to outperform both tree- and counting-based state-of-the-art algorithms [10].

### 3. EXPRESSION MATCHING MODEL

In our pub/sub matching model, the input is a set of subscriptions (a conjunction of Boolean predicates) and an event (an assignment of a value to each attribute), and the output is a subset of subscriptions satisfied by the event. In fact, we also model our events as Boolean expression to enable a more powerful event language.

In short, we define a Boolean expression as conjunction of Boolean predicates. A predicate is a triple, consisting of an attribute uniquely representing a dimension in  $n$ -dimensional space, an operator, and a set of values, denoted by  $P^{\text{attr, opt, val}}(x)$ , or more concisely as  $P(x)$ . A predicate either accepts or rejects an input  $x$  such that  $P^{\text{attr, opt, val}}(x) : x \rightarrow \{\text{True}, \text{False}\}$ , where  $x \in \text{Dom}(P^{\text{attr}})$  and  $P^{\text{attr}}$  is the predicate's attribute. Formally, a Boolean expression  $be$  is defined over an  $n$ -dimensional space as follows:

$$be = \{P_1^{\text{attr, opt, val}}(x) \wedge \dots \wedge P_k^{\text{attr, opt, val}}(x)\},$$

$$\text{where } k \leq n; \quad i, j \leq k, \quad P_i^{\text{attr}} = P_j^{\text{attr}} \text{ iff } i = j$$

We support an expressive set of operators for the most common data types: relational operators ( $<$ ,  $\leq$ ,  $=$ ,  $\neq$ ,  $\geq$ ,  $>$ ), set operators ( $\in$ ,  $\notin$ ), and the SQL BETWEEN operator.

Our subscription language is expressive enough to capture location-based information. The location information can be encoded as either longitude/latitude or three-dimensional  $(x, y, z)$  coordinates. For instance, a location in Toronto can be expressed as [Latitude = 43.6481] and [Longitude = -79.4042].

**Matching Semantics** The expressiveness of our subscription and event language enables supporting a wide range of matching semantics. In this paper, we focus primarily on the classical pub/sub matching problem: *Given an event  $\epsilon$  and a set of subscriptions  $\Sigma$ , find all subscriptions  $\sigma_i \in \Sigma$  satisfied by  $\epsilon$ .* We refer to this problem as *stabbing subscription*  $\text{SQ}(\epsilon)$  given by:

$$\text{SQ}(\epsilon) = \{\sigma_i \mid \forall P_q^{\text{attr, opt, val}}(x) \in \sigma_i, \exists P_o^{\text{attr, opt, val}}(x) \in \epsilon,$$

$$P_q^{\text{attr}} = P_o^{\text{attr}}, \exists x \in \text{Dom}(P_q^{\text{attr}}), P_q(x) \wedge P_o(x)\}$$

### 4. EVALUATIONS

Our evaluation is conducted using an Intel Xeon W3565 3.20GHz (launched in 2009) machine with 6GB of memory running Ubuntu 12.04. In our experiments two major algorithms were considered. Our PC-based algorithm, namely, BE-Tree, is single-threaded and written in C and compiled using gcc 4.6.3 [10]. In addition, we have drawn results from the GPU-based algorithm CLCB that is ran on Nvidia GTX 460 (launched in 2010-11) with 1GB of memory using the CUDA Toolkit 4.1 [6].

We experiment with two variations of BE-Tree, which are both based on [10]. BE-Tree 1.1 is based on [10] (and partial optimization from [11]), whereas BE-Tree 1.3<sup>1</sup> consists of all optimizations that were introduced in BE-Tree's extended paper [11].

We rely on default workloads used in [6], which serves as a common benchmark. The workload consists of executing 1000 events over 2.5 millions subscriptions. Both events and subscriptions consist of 3-5 predicates drawn uniformly from 100 dimensions, where the cardinality of each domain is 65K. The subscriptions and events in the location-based workload have only 2-4 regular predicates plus an additional location-based predicate. Since the location predicates generated in [6] have higher selectivity, the average number of matched subscriptions are increased from 3 to

<sup>1</sup>The BE-Tree 1.3 binary is available on <http://www.cs.toronto.edu/~mo/projects.html>

**Table 1: Comparing BE-Tree (PC) and CLCB (GPU)**

Workload Type	BE-Tree 1.1	BE-Tree 1.3	CLCB
without location	0.081 ms	0.045 ms	N/A
with location	0.144 ms	0.067 ms	0.306 ms

70 when moving from the workloads without to with location predicates.

The experimental results are summarized in Table 1. The CLCB algorithm (ran on GPUs) on average processes each event in 0.306 ms for location-based workloads while BE-Tree 1.3 can sustain an average matching time of 0.045 ms for non-location-based workload and matching time of 0.067 ms for location-based workload (nearly 5x faster than CLCB). The slight increase in BE-Tree matching is due to increased number of matched subscriptions.

Therefore, contrary to the reported results in [6] (based on BE-Tree 1.1), in fact, not only BE-Tree's matching time does not deteriorate when adding location-based information (because BE-Tree makes no distinction between location and non-location-based predicates and BE-Tree's underlying novel two-phase space-cutting technique can utilize all types of predicates), but BE-Tree also outperforms the CLCB algorithm ran on GPUs [5, 6]. The incorrect BE-Tree results reported in [6] was due to supplying BE-Tree with wrong parameters, not visible to the authors in [6]; thus, the authors were unaware of this misconfiguration [6].

### 5. CONCLUSIONS

In this work, we demonstrated that BE-Tree [10], a novel index structure to efficiently index Boolean expression over a high-dimensional space, can seamlessly support expressions with location-based predicates. In addition, we shown that the BE-Tree matching time using a single thread substantially outperforms the specialized matching algorithms designed for GPUs [5, 6].

### 6. REFERENCES

- [1] R. Agrawal, A. Ailamaki, P. A. Bernstein, et al. The Claremont report on database research. *SIGMOD Rec.'08*.
- [2] 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*.
- [3] F. Fabret, H.-A. Jacobsen, F. Llirbat, J. Pereira, K. A. Ross, and D. Shasha. Filtering algorithms and implementation for fast pub/sub systems. *SIGMOD'01*.
- [4] A. Farroukh, M. Sadoghi, and H.-A. Jacobsen. Towards vulnerability-based intrusion detection with event processing. *DEBS'11*.
- [5] A. Margara and G. Cugola. High performance content-based matching using GPUs. *DEBS'11*.
- [6] A. Margara and G. Cugola. High-performance location-aware publish-subscribe on GPUs. *Middleware'12*.
- [7] B. Ooi, K. Tan, and A. Tung. Sense the physical, walk through the virtual, manage the co (existing) spaces: A database perspective. In *SIGMOD Rec.'09*.
- [8] M. Sadoghi. Towards an extensible efficient event processing kernel. In *SIGMOD/PODS PhD Symposium'12*.
- [9] M. Sadoghi, I. Burcea, and H.-A. Jacobsen. GPX-Matcher: A generic Boolean predicate-based XPath expression matcher. In *EDBT'11*.
- [10] M. Sadoghi and H.-A. Jacobsen. BE-Tree: an index structure to efficiently match Boolean expressions over high-dimensional discrete space. In *SIGMOD'11*.
- [11] M. Sadoghi and H.-A. Jacobsen. Indexing Boolean expression over high-dimensional space. *Technical Report CSRG-608, University of Toronto, 2010*.
- [12] M. Sadoghi and H.-A. Jacobsen. Relevance matters: Capitalizing on less (top-k matching in publish/subscribe). *ICDE'12*.
- [13] M. Sadoghi, H.-A. Jacobsen, M. Labrecque, W. Shum, and H. Singh. Efficient event processing through reconfigurable hardware for algorithmic trading. *PVLDB'10*.