ISSN:0975-9646

Efficient Two-Phase Parallel Content Matching Algorithm for Publish-Subscribe Systems

Medha Shah[1] Suhas Doijad[2] Dinesh Kulkarni[3]
Walchand College of Engineering,
Sangli, Maharashtra, India.

***Abstract*— Content-based subscriptions systems are an emerging alternative to traditional Publish-Subscribe systems because they permit more flexible subscriptions along multiple dimensions. In these systems, each subscription is a set of predicates which may test arbitrary attributes within an event. However, the matching problem for content-based systems, determining for each event the subset of all subscriptions whose predicates match the event, is still an open problem. We present efficient, scalable tree based technique as well as the parallel implementation of it, and discuss their impact. The tree-based technique improves limitation of table based approach. Also, we present optimized two phase matching algorithm. Result shows 65% reduction in matching time, and increase in throughput by 82%.**

***Keywords* — Publish-Subscribe system; Two- phase matching; Trie; Parallel Search Tree; Clustering**

I. INTRODUCTION

Events are everywhere. New sources of events like social feeds, IoT devices, RFID tags, cameras, mobile devices, internet services, websites and data repositories generate events at an enormous rate. The amount of data is growing exponentially. Every day at least 2.5 quintillion bytes of data get produced. The growth in the size of data is exponential. Many applications want to exploit these events in real time. Many distributed applications use Pub-Sub communication paradigm as communication backbone. In the Pub-Sub model, subscribers typically receive only a subset of the total messages published by one or more publisher. Here receivers declare their interest in the particular event in the form of subscription. The publisher publishes the information of interest as message or notification. Content-based Pub-Sub delivers to the subscribers published messages, which match subscriber‘s declared interest. The key problem in the content-based pub-sub system lies in an efficient matching of an event against a large number of subscribers on a single message broker. To minimize user-perceived event delivery latency and to deliver high throughput are two fundamental goals of the Pub-Sub system.

Various sequential matching algorithms [2-5] are proposed earlier. As they are effective at increasing throughput and reducing the matching time they fail to exploit parallel architecture available in today’s generation of computers.

Sequential matching algorithms generally fall into one of two classes [9]. The first class consists of two-phase algorithms [1, 5], where predicates are evaluated in the first phase and matched subscriptions are computed in the second phase. The second class compiles predicates into a tree structure, where internal nodes represent predicates and leaf nodes represent subscriptions. Publications traverse the tree along the path of matching predicates ultimately leading to matching subscriptions (if any). In this paper, we parallelize the two-phase algorithm that makes use of Trie data structure, Parallel Search Tree (PST) and Table based approach in the first phase. Also, the more optimized approach has been suggested and tested in the second phase. Probably we are the first one to introduce tree based approach in the two-phase algorithm.

The rest of the paper is organized as follows: Section 2 presents the related work about Pub-Sub systems. Section 3 discusses the methodology used for two-phase matching algorithm along with three new implementations. Complexity is evaluated in Section 4 with parallelization strategy. Section 5 demonstrates experimental results. Finally, Section 6 provides some conclusive remarks and describes future work.

II. RELATED WORK

Many researchers have attempted to devise new matching algorithms for content-based Pub-Sub systems. Authors [2-4] have implemented different algorithms which are sequential one and amenable for parallelization. Most of the researchers have stick to two-phase algorithms due to its advantage in performance and storage. In [1, 3, 5, 9] authors have proposed a two-phase algorithm which makes use of the Table based approach. Two parallelization strategies have been implemented by authors in [1]. The algorithm presented in [3] is considered as a base for parallel implementation.

Several approaches for XML-based Pub-Sub are given in [11-13], in which different concepts like XFilter YFilter, BFilter and top-down matching, bottom-up matching are reported. XFilter [11] is based on deterministic finite automata (DFA), which stores user queries and handles each query individually. Yfilter emphasizes prefix sharing by using nondeterministic finite automata (NFA).

III. METHODOLOGY

A. Data Model

A subscription is a set of predicates, which are itself a tuple containing attribute name, operator, and value. An event is a collection of pairs, where each pair consists of an attribute name and value. The operator in a subscription can be <, ≤, ≠, =,> or ≥. An event's pair, (<attribute name> x, <value> y), matches a subscription's predicate (<attribute name> a, <operator> b, <value> c) only when x = a and y <operator b> c is true.

B. Two Phase Matching Algorithm

Here we present the optimized two phase matching algorithm given in [5]. Two phase algorithm operates in two phases namely H-phase and C-phase. More efficiency is achieved by introducing counting along with clustering in the second phase. Clusters are formed based on a number of predicates in a subscription. The key to accessing the cluster is the number of predicates in an event. The following figure depicts a combination of both, counting and clustering approach. The figure 1 shows table based approach in the first phase and counting along with clustering in the second phase.

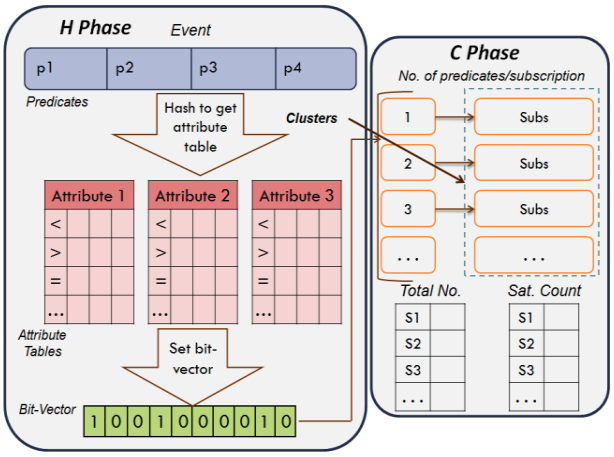

Fig. 1 Two Phase Matching Engine

The advantage of using table based predicate data structure is i) easy to access ii) small overhead for storing iii) spatial locality. But there are some limitations like i) size of the table has to be predefined ii) domain type of value is restricted to an integer. Trie and Parallel Search Tree [6-8] can be used to overcome the limitations. The following section describes these data structures in detail.

C. Trie-Based Data Structure

Trie is tree data structure used to store strings. A popular implementation of Trie is dictionary search. Strings can be easily stored in Trie, but for storing a predicate, some modification is required. The whole predicate is considered as a string and stored it into Trie. Figure 2 shows Trie implementation of some of the predicates.

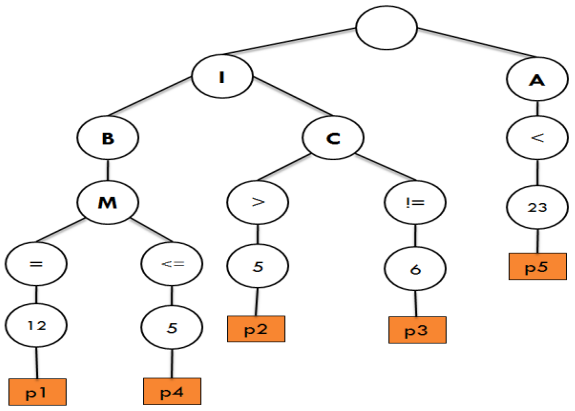

Fig. 2 Trie Data Structure

The advantage of using this is, all possible domains can be included in the predicate. But space required for this approach is much more. To reduce space overhead another approach, parallel search tree (PST) can be used.

D. Parallel Search Tree-Based Data Structure

The *Gryphon* project [8] uses a PST algorithm to solve the matching problem in Pub-Sub systems. In PST each node corresponds to a test and each subscription is a path from the root to a leaf. Given an event e, it matches all subscriptions reached by a tree traversal that only follows an edge if e matches the constraint denoted by the attribute of the level, followed by the node and edge labels. Intuitively, the data structure factorizes tests common to several subscriptions and thus favors scalability, since it allows a sub-linear matching complexity [7] with respect to the number of different subscriptions. The following figure depicts a simple PST.

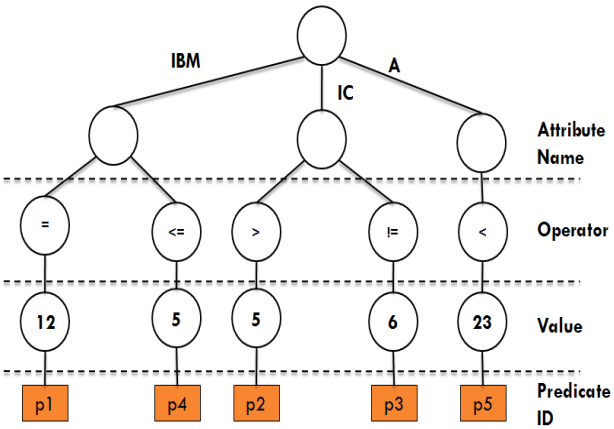

Fig. 3 PST Data Structure

Looking up data in a Trie data structure, PST is faster, compared to an imperfect hash function. So hashing phase from original two- phase algorithm is replaced by tree formation (T Phase) as shown in the following the figure. Figure 4 shows modified two-phase algorithm shown in figure 1 with tree data structure.

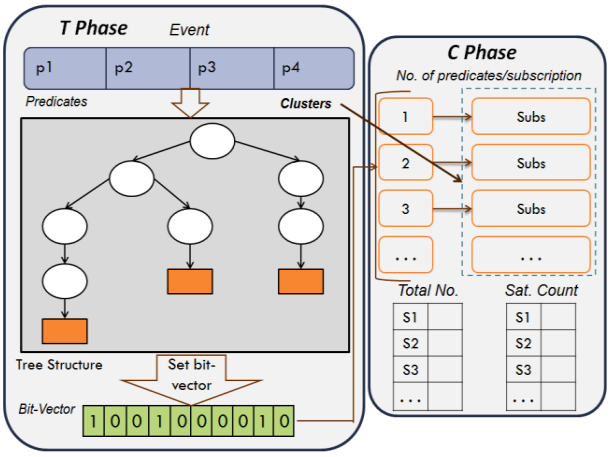

Fig. 4 Tree Based Two Phase Matching Engine

The whole two-phase matching process will look as shown in figure 5.

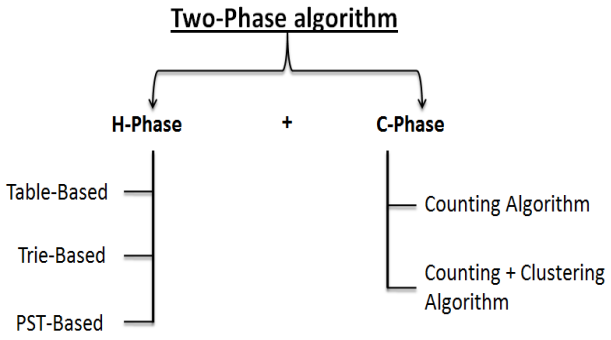

Fig. 5 Overview of Two – Phase Algorithm

IV. EVALUATION

We have implemented the two-phase algorithm using three different data structures in H-phase. Both, counting and counting along with clustering is implemented in C-Phase. For experimentation purpose workload is generated as discussed in the paper [1, 9]. Workload contains 100 different attribute names with more than 1500 different predicates. The number of subscriptions ranges from 2000 to 100000 along with 2000 events. Different parallelization techniques are presented in [1]. ME-IP (Multiple Events Independent Processing) technique as discussed in [1] is used to increase throughput (i.e. number of events processed per second) of the system.

A. Pre-Processing Time Complexity

Pre-processing time [6] is the time to form data structure based on input subscriptions. For N subscriptions and P unique predicates, the pre-processing time is almost same for all three data structures, which is O (NP).

B. Space Complexity

For storing N subscriptions table-based data structure requires O (A) space, where A is no. of unique attributes. Whereas for Trie, space required is O (W), where W is the total length of P predicates and for PST based it is O (P).

C. Matching Time Complexity

As counting along with clustering is used in C-Phase, the time required for getting matched subscription is NP_{sat} * S_{avg} + N, where, NP_{sat} = No. of predicates satisfied by an event, S_{avg} = Avg. no. of subscriptions per cluster.

V. RESULT AND ANALYSIS

All experimental results are taken on 8 cores Intel Core i7-2600 CPU running at 3.40 GHz. The operating system is Cent-OS with kernel version 2.6.32. Compiler used is GCC 4.4.6.

A. C Phase Implementation

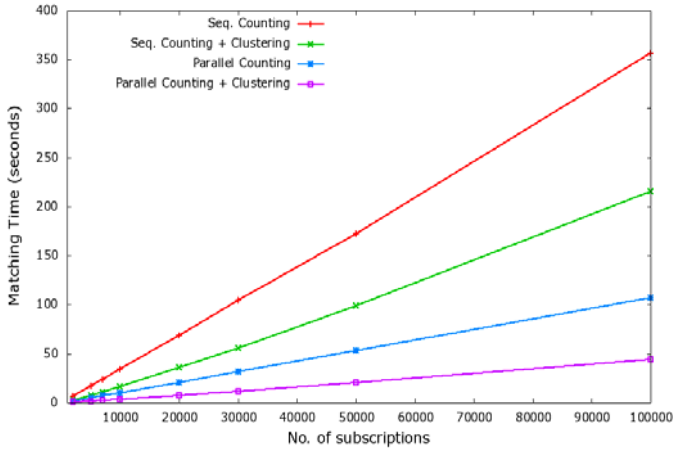

Fig. 6 Comparison of matching performance (Counting and Counting with Clustering)

We first look at the performance of two algorithms used in C- Phase. Figure 6 shows time required for matching subscriptions for counting and counting with clustering approach. Table Based data structure is used in the first phase. Results shows, counting along with clustering is better than counting, with improvement in matching the time of 65%.

B. Input Size versus Matching Time

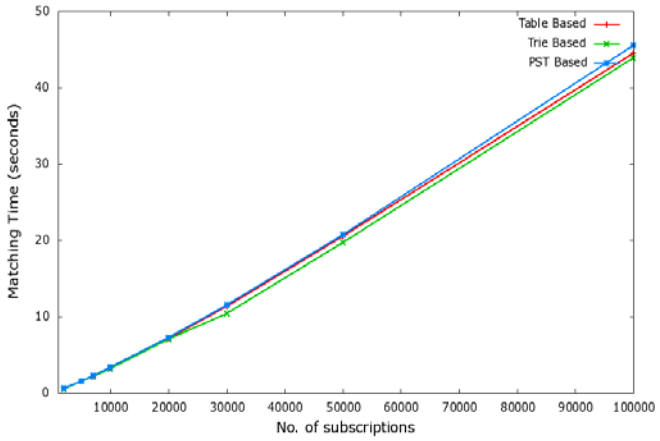

Fig. 7 Subscription matching performance

Figure 7 shows time required for parallel (ME-IP) implementation of three approaches. Trie approach is more efficient than table or PST. So, as the system has multiple cores, multiple events are processed simultaneously, which increases system's throughput.

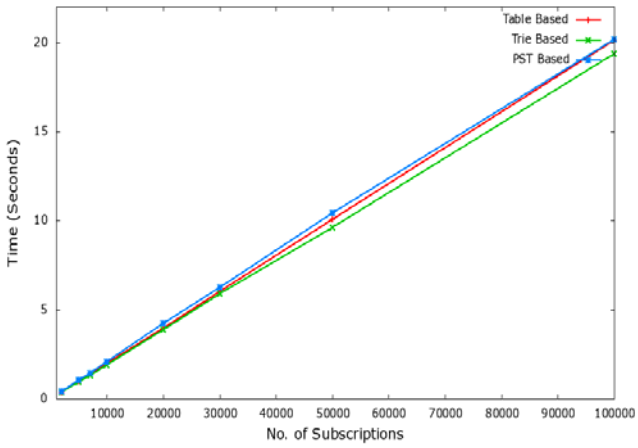

Fig. 8 Time for pre-processing

Figure 8 shows pre-processing time required for Table, Trie and PST approach. This time contains, the time required to build an actual structure which holds all the predicates along with the creation of clusters and mapping of predicates to subscription. Less time is required to build the Trie data structure as compared with other two data structures.

C. *Input Size versus Storage Space*

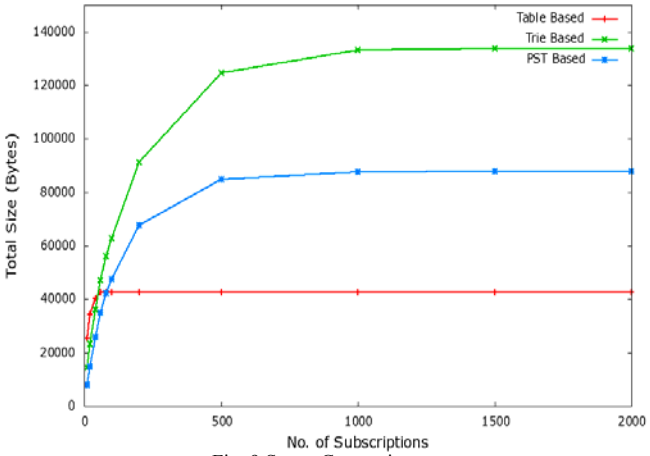

Fig. 9 Space Comparison

Figure 9 shows space requirement of all three approaches. For Table Based approach size depends upon how many tables are created at run time. Whereas for Trie based and PST based structures, space required depends on a number of predicates. Hence space requirement will go on increasing until unique predicate occurs. Trie data structure is more space consuming as compared to PST because Trie stores attribute name in a predicate character by character. Whereas, PST stores whole attribute name in a single node of the tree.

D. *Processing Units versus Matching Time*

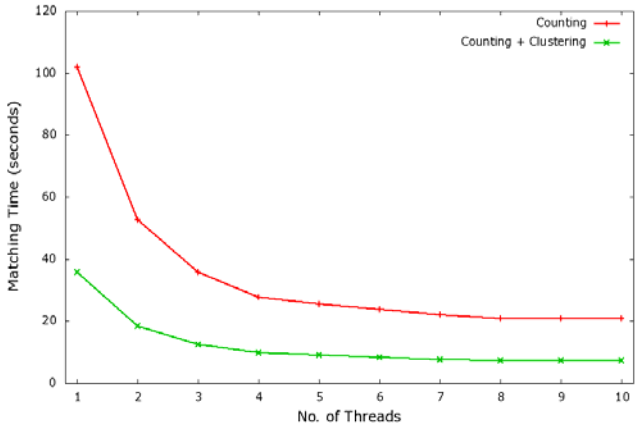

Fig. 10 Matching Time requirement of processing units

Figure 10 shows matching time required for a different number of threads. As a number of threads increases, matching time decreases. But after certain number (i.e. system's core limit) time remains constant. The figure compares matching time requirements of two C-Phase algorithms.

E. *Throughput*

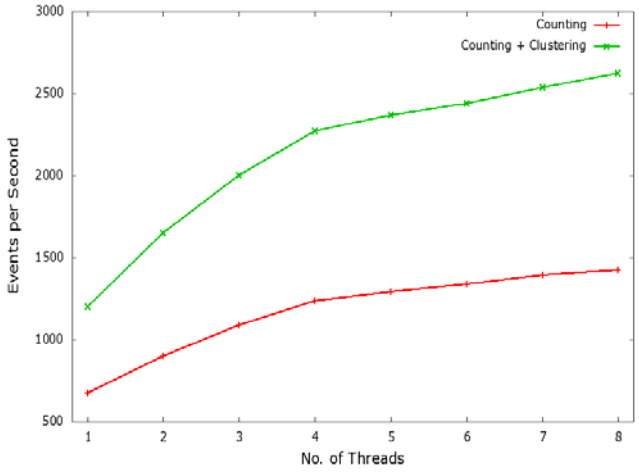

Fig. 11 Throughput of ME-IP for two approaches

Figure 11 shows throughput (ME-IP) of the system using two different algorithms for C-Phase. As counting along with clustering, reduces the matching time, a number of events are processed compared to simple counting. Almost 82% more events are processed with combined approach.

VI. CONCLUSION

In this paper, we have proposed Trie and parallel search tree data structure to be used in H-phase of the two-phase algorithm. We also suggest counting + clustering approach be used in C-phase of the two-phase algorithm. We presented space and time complexity for approaches mentioned. To increase the throughput of the system, we have used ME-IP parallelization technique. Time for pre-processing and matching is almost same for all three approaches. Table based approach requires less space to store predicates. But we can't include other data

types such as float or string. This limitation is removed in two tree-based techniques. The trie-based approach is slightly more efficient than PST in terms of matching time. But it requires more space than that of PST. On average PST is suitable for all cases. In the second phase, counting with clustering outperforms simple counting. In future, we will implement different techniques to reduce the matching time and improve efficiency.

REFERENCES

[1] A. Farroukh, E. Ferzli, N. Tajuddin, and H. Jacobsen "Parallel event processing for content-based publish/subscribe systems," in proceedings of the 3rd ACM International Conference on Distributed Event-Based Systems, 2009.

[2] Françoise Fabret , François Llirbat , João Pereira and Dennis Shasha, "Efficient matching for content-based publish/subscribe systems ",in 2000.

[3] F. Fabret, H.-A. Jacobsen, D. Shasha st.al., "Filtering algorithms and implementation for very fast publish/subscribe systems", in SIGMOD, 2001.

[4] P. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," ACM Comput. Surv., vol. 35, pp. 114–131, June 2003.

[5] G. Ashayer, H. K. Y. Leung, and H.-A. Jacobsen "Predicate matching and subscription matching in publish/subscribe systems," In Proceedings of the 22nd International Conference on Distributed Computing Systems, ICDCSW '02, pages 539–548, Washington.

[6] Marcos K. Aguilera, Robert E. Strom et.al ,"Matching events in a content-based subscription system".

[7] J. Legatheaux Martins, S´ergio Duarte, "Routing algorithms for content-based publish/subscribe systems", 2008.

[8] P. Jayachandran, R. K. Ganti et.al ,"Benefits of inter-tree optimizations for content based publish-subscribe in sensor setworks".

[9] Zhaoran Wang, Yu Zhang, et.al,"Pub/Sub on Stream: A multi-core based message broker with QoS support",in DEBS 2012, July 16–20, 2012, Berlin, Germany July 2012.

[10] Patel Kuldip L. , Savalia Jay M, et.al "Parallelization of complex event processing on GPU",HiPC in 2011.

[11] L. Dai, C.-H. Lung, and S. Majumdar. "BFilter – A XML message filtering and matching approach in publish/subscribe systems", in Proc. of the IEEE GLOBECOM, Dec. 2010.

[12] Roger Moussalli, Vassilis J. Tsotras, et.al, "Efficient XML path filtering using GPUs",in 2nd International Workshop on Accelerating Data Management Systems using Modern Processor and Storage Architectures (ADMS"11), 2011.

[13] Yang Cao , Chung-Horng Lung , Shikharesh Majumdar, "A peer-to-peer model for XML publish/subscribe services" in 9th Annual Communication Networks and Services Research Conference, IEEE 2011.