Parallel Content Matching in Publish or Subscribe Systems

Suhas Doijad[1], Medha Shah[2]

[1]Dept. of Computer Science and Engineering, Walchand College of Engineering, Sangli, Maharashtra, India
[2]Dept. of Computer Science and Engineering, Walchand College of Engineering, Sangli, Maharashtra, India

Abstract: ***Publish/Subscribe (Pub/Sub) system is the more optimized implementation of complex event processing system, in which user gives the interest (subscriptions) and some other party publishes the event (e.g. Stock quotes). The main functionality of Publish/Subscribe system is to send these events to subscribers whose subscriptions are related to the events. So the core of this system is to match the events with the subscriptions. As two phase algorithm is good in spatial locality and performs better, its implementation is proposed in this paper. The process of matching is parallelized using current cheaper and commonly available multi-core systems. Also, different parallelization strategies are used to improve the throughput and to reduce the matching time. Performance of the system is measured by taking processing time as a parameter. The results show that, the proposed technique gives more throughputs along with speedup of 3.4x and efficiency of 40%.***

Keywords: Publish/Subscribe system, Multi-core processors, Matching algorithm, Event processing, Stock quotes.

1. Introduction

The Publish/Subscribe system deals with filtering of each event against the subscriber's subscription. A subscription shows subscriber's interest related to a specific event and publication resembles these events. The Publish/Subscribe system performs the matching of events to that of subscriptions and notifies the users according to their interest. In this model both subscribers and publishers are decoupled and connected only through the Publish/Subscribe system (Message broker OR Middleware). Fig 1 shows a scenario of Publish/Subscribe system (Stock Quotes). In this, subscribers are the shareholders which want information about their shares. Publishers are the stock providers (e.g. BSE).

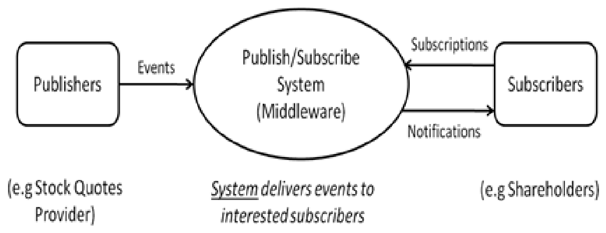
Publishers
Events
Publish/Subscribe System (Middleware)
Subscriptions
Subscribers
Notifications
(e.g Stock Quotes Provider)
System delivers events to interested subscribers
(e.g Shareholders)

Figure 1: Publish/Subscribe System in stock quotes

Events are a set of predicates. Each predicate is a pair in the form <attribute = value>. Similar to events, subscriptions are also set of predicates in the form <attribute operator value>. An event's pair, (<attribute>p, <value> q); matches subscription's predicate (<attribute> x, <operator> y, <value> z) only when p=x, and q <operator y> z.

There are two categories of Publish/Subscribe system, one is topic-based and another is content-based. Attribute is used for matching in topic-based Publish/Subscribe system while value is used in content-based system. So, matching in content-based system is more compute intensive task which can be done with the use of recent multi-core processors commonly available along with parallel programming constructs like OpenMP.

The rest of the paper is organised as follows: Section 2 presents the related work about Publish/Subscribe systems. Section 3 discusses the matching process along with parallelization strategies. Section 4 demonstrates experimental results. Finally section 5 concludes this paper.

2. Related Work

Several researchers have attempted to improve the performance of content based Publish/Subscribe system through parallelism [1], [2]. In [1] authors have implemented parallel matching engine which is given in sequential form in [3]. Many sequential algorithms are designed for content based matching in [3], [4], [5] and [6]. Some of them are amenable for parallelization. In [2] authors have shown parallel implementation for Job Portals using threads.

Several approaches for XML message filtering for pub/sub systems are given in [7], [8] and [9]. In [7] authors exploit the parallelism found in XPath filtering systems using GPUs. Current XML filtering research work could be classified into three classes: top-down matching, bottom-up matching and sequence-based matching. Different filters such as Yfilter and Bfilter are reported in the literature [8] and [9].

The main functionality of Publish/Subscribe systems is the matching algorithm. There are two types of matching algorithms - two-phase algorithm and compilation method. Two phase algorithm [5] works in two phases. First, predicates in the events are evaluated against all the predicates in all subscriptions. This forms an intermediate result. Second, this intermediate result is used to compute matched subscriptions. In compilation method, matching algorithm forms a tree-like structure to match events with all subscriptions. We use two-phase algorithm because it's simple storage and high performance.

3. Matching Process

3.1 Sequential Implementation

The Naive Algorithm is given in [4]. In this, each event is evaluated against all subscriptions sequentially. A more optimized algorithm, based on two-phase is presented in [1] and [5]. In first phase (H Phase), table based data structure [5] is used to store all predicates in subscriptions where as in second phase (C Phase), counting is done. Counting is a most efficient algorithm which does not require any additional information [5]. The counting algorithm counts the number of predicates satisfied by an event and checks whether they are equal to total predicates in a subscription(match) or not.

Another method that can be used in second phase is clustering [3]. For clustering some additional information such as statistics on predicate is required for making clusters. So group of subscriptions will form a cluster, which is accessed by an access predicate. If the access predicate is satisfied then only the cluster associated with it will be checked for its matching. So clustering depends upon how the access predicate is chosen and number of subscriptions which are associated with that access predicate. Generally equality predicate (a predicate with '=' operator in a subscription) acts as an access predicate, because it's easy to check whether equality is achieved or not.

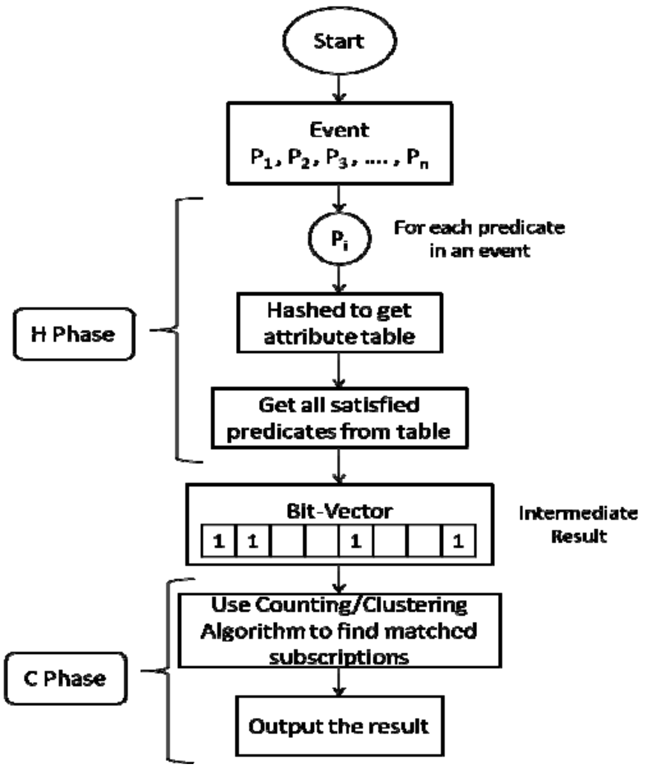
Start
Event
$P_1, P_2, P_3, \ldots, P_n$
P_i
For each predicate in an event
H Phase
Hashed to get attribute table
Get all satisfied predicates from table
Bit-Vector
1 1 1 1
Intermediate Result
C Phase
Use Counting/Clustering Algorithm to find matched subscriptions
Output the result
Figure 2: Two-phase matching Process

Fig. 2 shows the whole process. Event is matched with the subscriptions in two phases (H phase and C Phase). H Phase processes each predicate in an event. Hashing is used to get an access of particular table. Each attribute is having a separate table (as show in fig. 3) which stores all predicates related to that attribute.

All satisfied predicates will be fetched and related bits in a bit-vector are set to '1'. After processing all predicates in an event, counting/clustering algorithm is used in C-Phase to get all satisfied subscriptions.

For counting two tables are maintained. First is to store number of predicates in a subscription and second is to store satisfied predicates by an event. Finally both tables are checked to get satisfied subscription/s by an event. But in case of clustering, access predicate is checked, whether it is satisfied or not with the help of bit vector. If a bit is '1' for an access predicate then it is satisfied, so the cluster associated with that predicate is processed.

Fig. 3 shows the table based approach. In this, separate table is made for each attribute occurred in subscriptions. Rows are for different operators and columns are for values related to that operator.

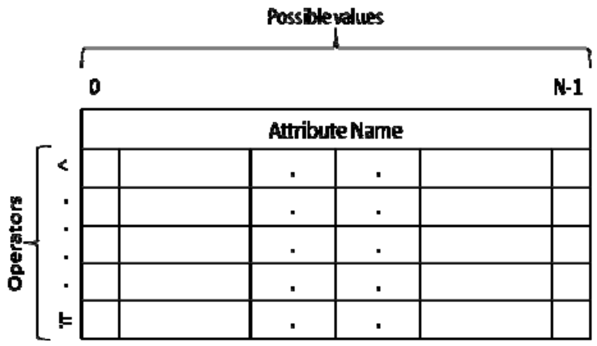

Figure 3: Table Based Data Structure

3.2 Parallel Implementation

Two parallelization strategies are given in [1]:

3.2.1 Single Event Collaborative Processing (SE-CP):

The matching time per event is reduced using SE-CP implementation. In this, event is fragmented according to predicates as a part and each part is evaluated separately.

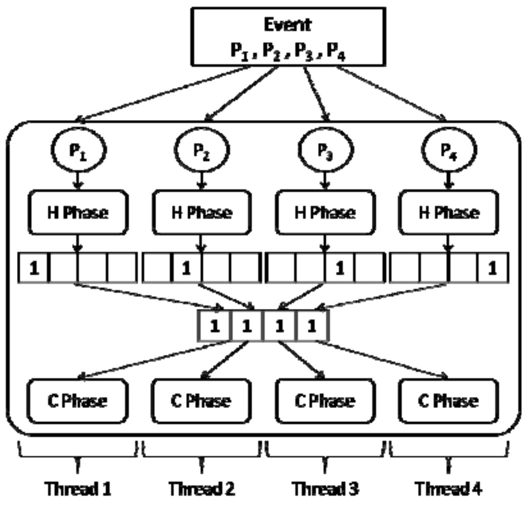

Figure 4: SE-CP Implementation

Fig. 4 describes the working of SE-CP. For every event, each thread takes a predicate and processes it and set bits in bit-vector. After completing H-Phase all threads update the global bit-vector. This global vector is further processed in

C-Phase by each thread separately. Synchronization is required to carry out the updating of global bit-vector.

3.2.2 Multiple Event Independent Processing (ME-IP)
This strategy is used to increase the system's throughput. As multiple events are processed simultaneously, more and more events are processed in less amount of time.

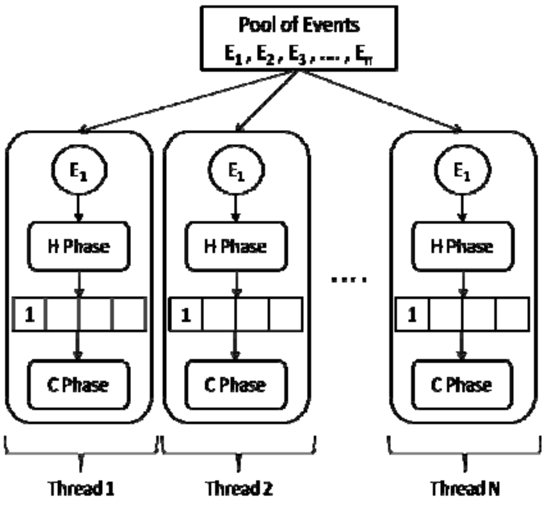

Figure 5:ME-IP Implementation

4. Experimental Results

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.

Workload for the experimentation purpose is generated as given in [1]. Both subscriptions and events are input to the system. First subscriptions are loaded and then events. Total 100 different attributes along with more than 1450 different predicates are used to generate subscriptions. The operators taken are < (less than), ≤ (less than equal to), > (greater than), ≥ (greater than equal to), = (equal to), and ≠ (not equal to). The events are generated accordingly.

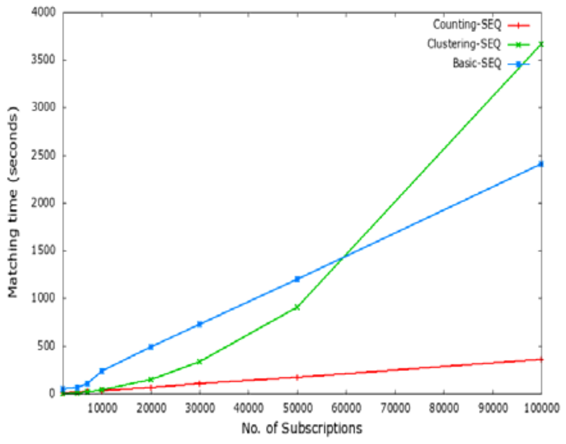

Figure 6: Matching time of sequential implementation

Fig. 6 shows the matching time required to match all the subscriptions against the events for three sequential algorithms. The naïve (Basic) algorithm performs matching of events with subscriptions one by one. So time taken for this approach is more as compared to two-phase algorithm using counting. But for clustering in two-phase, time required is more as the cluster size increases. The number of subscriptions ranges from 5000 to 100000 along with 2000 events.

Fig. 7 shows the comparison of sequential and SE-CP implementation. SE-CP technique depends upon how many predicates are there in an event. So graph is plotted taking number of predicates on x-axis versus time to match that event with subscription on y-axis. The no. of predicates ranges from 1 to 10.

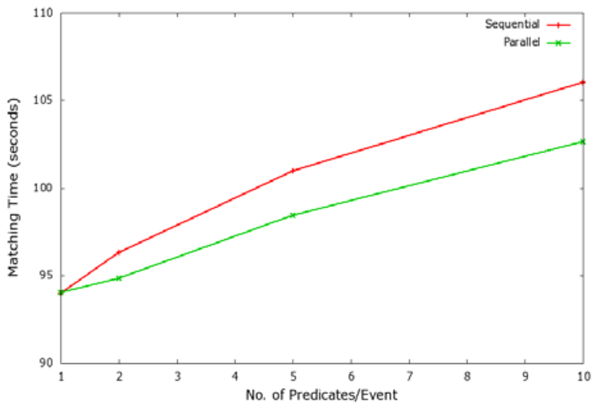

Figure 7: SE-CP Implementation

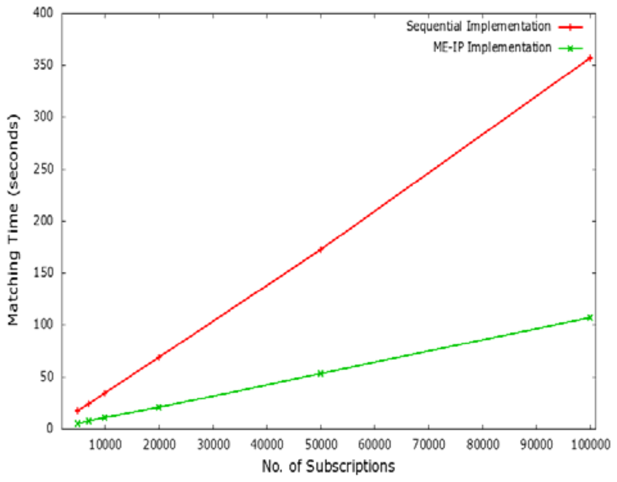

Figure 8: ME-IP Implementation

Fig. 8 represents time required for sequential and parallel implementation of two-phase algorithm. ME-IP implementation is considered here. So, as system has multiple cores, multiple events are processed simultaneously increasing system's throughput.

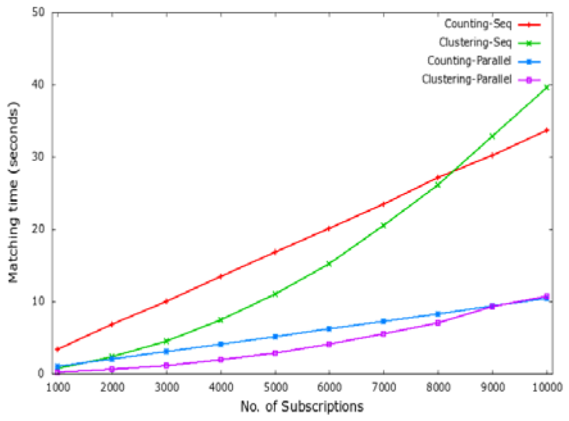

Figure 9: Comparisons of counting and clustering algorithm

Fig. 9 shows difference between counting and clustering algorithm for 1000 to 10000 number of subscriptions. Initially clustering shows better performance than counting. This is because initially number of clusters and size of each cluster are less, so matching time is also less. But as the number of subscriptions increases both, number of clusters and size of each cluster, increases.

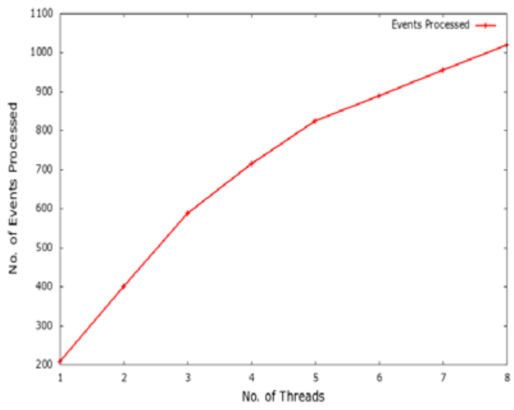

Figure 10: Events processed per second

Fig. 10 shows the throughput of ME-IP. As the number of threads used in a system increases, more number of events is processed in a second. So there is a linear increase in throughput. For a single thread, around 200 events are processed per second whereas for 8 threads (default number of threads in i7 processors with multi-threading) 1019 events per second are processed.

5. Conclusion

In this paper, we have presented table based approach to store the subscriptions for its high performance. Also, two parallelization strategies (SE-CP and ME-IP) which are useful in Publish/Subscribe systems are presented. ME-IP increases throughput whereas SE-CP improves matching time. The results show that, use of 8 core system increases throughput from 200 to 1019 events per second. There are two algorithms which can be used, i) counting and ii) clustering for final matching. The performance of clustering algorithm depends upon the way clustering is done along with size of the clusters. In future we will focus on implementing other approaches for storing subscriptions which are amenable to parallelism.

References

[1] A. Farroukh, E. Ferzli, N. Tajuddin, and H. Jacobsen "Parallel event processing for content-based Publish/ Subscribe systems," In Proceedings of the Third ACM International Conference on Distributed Event-Based Systems, page 8. ACM, 2009.

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

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

[4] Françoise Fabret , François Llirbat , João Pereira et.al., "Efficient Matching for Content-based Publish/Subscribe Systems ",in 2000.

[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] P. T. 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.

[7] Roger Moussalli, Vassilis J. Tsotras, et.al, "Efficient XML Path Filtering Using GPUs",in Second International Workshop on Accelerating Data Management Systems using Modern Processor and Storage Architectures (ADMS"11), 2011.

[8] 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.

[9] 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.

Author Profile

Suhas Doijad received B.E. degree in Computer Science and Engineering from Shivaji University, Maharashtra, India, in 2011. Currently pursuing Masters in Computer Science and Engineering from Walchand college of Engineering, an autonomous institute, Maharashtra, India.

Medha Shah is an Assistant Professor in the Department of Computer Science and Engineering at Walchand College of Engineering, Sangli, Maharashtra, India. She has completed M.E. in Computer Science and Engineering in 2008. She is currently pursuing Ph.D in Computer Science and Engineering from Walchand College, Sangli, Maharashtra, India.