**Parallelized Event Matching in Content-based Publish/Subscribe System**

**Abstract**: Content-based event matching is an important problem in event based pub/sub system. There are many pub/sub system like JMS, Corba NS. Siena is research project for designing and constructing generic scalable pub/sub event notification service. This project makes a few contributions toward analysis, evaluation and development of SIENA distributed system and SIENA fast forwarding matching algorithm. We analyzed matching time taken by existing matching algorithm. To reduce matching time we developed the algorithm by using parallelism.

**Keywords:** publisher, subscriber, subscription, content-based, matching time.

**Introduction:** In pub/sub system subscribers are the users who give their choice in the form subscriptions and providers of information publish events called publishers. Broker also called server matches subscriptions with published events in filtering process. Respective events are sent to interested subscribers. There are two main forms of filtering. First one is Topic based, messages are published to "topics" or named logical channels. Second is content based, where messages are received to subscriber only if the attributes or content of those messages match constraints given by the subscribers. SIENA is content based pub/sub system and focuses on distributed architecture. SIENA is free software, first released in 1998. It is available in both C++ and Java language. We need to set up server to connect to publisher and subscriber.

SFF is counting algorithm used for event matching and focus is on predicate matching. Message matches constraints if it contains an attribute with the same name and value, and if the value matches the condition specified by subscriber.

**Term Definitions:**

**Events:** An event, also called message or publication is conjunction of attribute value pairs published by publisher. Ex. M= {a1, a2…, am} where M is message and a1, a2…, am are attributes, number of attributes is denoted m.

**Primitive Constraints**: Primitive constraint is a condition given by subscriber on an attribute. For Ex. C={less than, greater than, equal to, not equal to, any}.

**Subscription:** subscription is the expression who gives the interest of subscriber.

**Event Matching**: Given a set of n subscriptions S= {s1, s2, s3 , …, sn} and an event e, event matching is to find all subscriptions from S that match e.

**Matching Time**: Matching time is defined as time running time needed by matching algorithm to match event against subscriptions.

**Background:**

SIENA (Scalable Interface Event Notification Architecture) system contains two main entities. clients and servers. The clients are of two types. Publishers and subscribers.

|  |
| --- |
|  |

Fig.1 pub/sub system

Publisher use access points to publish the notifications and subscribers use access points to subscribe the notification of interest by supplying predicate called filter. SIENA is the notification data model provides the service. Notification contains set of typed attributes. Each attribute has a type, a name, and a value. The primitive constraints are used in the filters provided by subscribers.

|  |  |  |
| --- | --- | --- |
| Type | Name | Value |
| string  Int  double | name  id  pointer | Abc  123  8.2 |

Fig. 2(a) SIENA notification

|  |
| --- |
| String name=Abc  double pointer>7.5 |

Fig. 2(b)

**Related Work**:

Matching algorithm: SFF is content based counting matching algorithm. It works in sequential algorithm. For better performance and to reduce matching time, we developed existing algorithm by using parallelizing it. For parallelization, executor services are used.

|  |
| --- |
| Algorithm Counting  //Input=Message m  //Output=Matched Message  Map<filter, int>counters set to null  Set<interface>matched set to null  //use parallelism by using executor service  //for n number of constraints, n executor //services used.  {  Add matching\_constraints(a) to the set <constraints> C  For (each c in C)  {  For (each f in c.filters)  {  If (constraint not matches)  {  If no of constraints not equal to counter  No increment in counter  }  Increment counter  If(size of counter equal to size of filter)  //Message matched  Output Message m  }  } |

SIENA Fast Forwarding algorithm is implemented in sequential manner. It matches each individual attribute sequentially with constraints specified by subscriber. Due to this, it takes more matching time with increasing number of subscriptions. To improve the matching time, we focus on parallelization of matching algorithm. For parallelization, we use Executor service packages. Packages of executor service are in-built in java. Matching is defined by covering relations

|  |
| --- |
| name=XYZ name=XYZ  id=1 id=5  pointer=8.5 pointer>7  pointer<9 |

It provides better result than the SFF matching algorithm. SFF contains more computation than the above algorithm.

**Evaluation:**

If we use Siena Fast Forwarding algorithm for event matching, following results found:

Fig. 3 graph of constraints vs time

In this system for each attribute it matches with every constraint. If attribute matches with constraint it increases the count and when number of constraints is exactly equal to count then message matches with subscription. In sequential Algorithm for n number of constraints loop executes n times. So with increasing no. of constraints matching time increases. As shown in figure matching time for parallel algorithm for same no. of constraints is less than that of sequential algorithm. But in parallel algorithm time remains constant for any no. of constraints because for n no. of constraints n executor services are used and they work in parallel manner. So even if we increase no. of constraints there is slight change in matching time which is negligible.

Fig. 4 graph of subscriptions vs time

SIENA pattern follows conjunction of constraints and disjunction of subscriptions. For sequential algorithm time increases with increase in no. of subscriptions. Because if we increase the no. of subscription that may increase in no. of constraints. So matching time increases. In the above graph matching time for parallel algorithm is less than that of Sequential algorithm. Because in parallel algorithm executor services are used for each no. of constraint.

**Conclusion:**

In this paper, we have presented our experience in analyzing the SIENA distributed system. SIENA is open source. It is free to change behavior of system. Our experience focuses on issues: understanding working of SIENA, studying and analyzing matching algorithms, developing matching algorithm for better performance. We developed existing SFF algorithm by parallelizing it. It gives better performance. So, we provided a formal description of SIENA and existing matching algorithms. Future work on the providing prioritized and deadline aware matching algorithm.

**Reference:**

1. Shiyou Qian, Jian Cao, Fr\_ed\_eric Le Mouel, Minglu Li, Jie Wang. Towards Prioritized Event

Matching in a Content-based Publish/Subscribe System. ACM. 9th ACM International Conference

on Distributed Event-Based Systems (DEBS'2015), Jun 2015, Oslo, Norway.

2. Mauro Caporuscio, Paola Inverardi, Patrizio Pelliccione . Formal Analysis of Clients Mobility in the SIENA Publish/Subscribe Middleware

3. Antonio Carzaniga, Alexander L. Wolf. Forwarding in a Content-Based Network

4. Satyen Kale, Elad Hazan, Fengyun Cao, Jaswinder Pal Singh. Analysis and Algorithms for Content-based Event Matching.

5.www.inf.usi.ch/carzaniga/SIENA/software/java2/index.html

6.www.inf.usi.ch/carzaniga/SIENA/software/java2/index.html

7.www.inf.usi.ch/carzaniga/SIENA/software/cpp/index.html

8.www.inf.usi.ch/carzinaga/SIENA/forwarding/index.html

9.www.inf.usi.ch/carzinaga/SIENA/forwarding/doc/index.html