****

**WALCHAND COLLEGE OF ENGINEERING, SANGLI**

**(AN AUTONOMOUS INSTITUTE)**

**Mega Project Report on**

--------------------------------------------------------------------------------

**Parallel REIN: Fast Event Matching Approach for Content Based Publish/Subscribe Systems**

--------------------------------------------------------------------------------

*Submitted By:* *Project Guide:*

ApoorvaArunKokadwar(2012BCS010) Mrs. M. A. Shah

PriyankaPopatraoBhogade(2012BCS033)

PratibhaBhimraoAlande(2012BCS054)

*A project report submitted in fulfillment of the Mega Project*

*For the degree of Bachelor of Technology*

At

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Year

**2015-16**

****

**WALCHAND COLLEGE OF ENGINEERING, SANGLI**

**(AN AUTONOMOUS INSTITUTE)**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**CERTIFICATE**

*This is to certify that, The Project Report entitled,*

**Parallel REIN: Fast Event Matching Approach for Content Based Publish/Subscribe Systems**

*Submitted By:*

ApoorvaArunKokadwar(2012BCS010) PriyankaPopatraoBhogade(2012BCS033) PratibhaBhimraoAlande(2012BCS054)

*is a bonafide record of their own work performed out by them in fulfillment of the final*

*yearB.Tech project in Computer Science and Engineering as specified in the*

*curriculum prescribed by Walchand College of Engineering,Sangli.*

**Mrs. M. A. Shah Dr.B.F.Momin**

*Guide**Head of Department*

**DECLARATION**

We, the undersigned, hereby declare that the project report entitled,

**Parallel REIN: Fast Event Matching Approach for Content Based Publish/Subscribe Systems**

Written and submitted by us to Department of Computer Science and Engineering,

Walchand College of Engineering, Sangli as a complete fulfillment for Mega Project under the guidance of Asst. Prof. M.A.Shah is our sincere work. The empirical results in this project report are based on the data collected by us. We understand that any type of copying is liable to be punished as the authorities deem fit.

Date :-

Place :-Sangli

ApoorvaArunKokadwar(2012BCS010)

PriyankaPopatraoBhogade(2012BCS033)

PratibhaBhimraoAlande(2012BCS054)

**ACKNOWLEDGEMENT**

We are pleased to present this project report entitled `Parallel REIN: Fast Event Matching Approach for Content Based Publish/Subscribe Systems ' to our college as a part of academic activity. We are very thankful to our project guide Mrs. M. A. Shah. We are very thankful to all our colleagues andthose who helped us directly or indirectly throughout this project work. It won't be fair if we fail to acknowledge our family members, because of whom we are here. Their enthusiasm and backing have really boostered our confidence which has enabled us tocomplete our project works.

ApoorvaArunKokadwar(2012BCS010)

PriyankaPopatraoBhogade(2012BCS033)

PratibhaBhimraoAlande(2012BCS054)

**ABSTRACT**

The project focuses on use of parallelization in REIN (Rectangular Intersection) which is a fast event matching algorithm for large scale content-based publish/subscribe systems.

The main idea behind REIN is to quickly ﬁlter out unlikely matched subscriptions rather than to determine whether a subscription is matched or not by counting its satisﬁed component constraints.

The proposed idea is to minimize the time consumption for event matching by system, using the OPENMP for parallelization of REIN event matching algorithm.

In experimental results,it is observe that parallel REIN has a better matching performance than serial REIN. The event matching speed is faster by an order of magnitude when the selectivity of subscriptions is high and the number of subscriptions is large.

**Contents**

Declaration i

Acknowledgement ii

Abstract iii

Contents iv

List of Figures vi

1 INTRODUCTION 1

1.1 Purpose 1

1.2Concept of Content Based publish subscribe system 1

1.3.How does REIN work? 4

2 Literature Survey 5

2.1 Study of Currently Existing System 5

2.2 Drawbacks of Existing System 6

2.3 Proposed System 6

2.4 Advantages of Proposed System 7

3 PROJECT PLANNING 8

3.1 Project Development Approach 8

4 SYESTEM DESIGN 11

4.1 Requirement Specification 11

4.1.1 Optimal Software Requirements 11

4.1.2 OPENMP 12

4.2 UML Diagrams 13

4.2.1 Use case Diagram of a system 13

4.2.2 Sequence Diagram of a system 14

4.3 System characteristics 15

4.4 Attributes 15

4.4.1 Usability 15

4.4.2 Availability 15

5 PROJECT OVERVIEW 16

5.1Overview 16

5.2 Definition:REIN 17

5.3 Index Structure 18

5.4 Event Matching 18

6 RESULTS 21

7 PROJECT OUTCOMES AND FUTURE SCOPE 24

7.1 Outcomes 24

7.2 Future Scope 24

8 CONCLUSION 25

9. BIBLIOGRAPHY 26

**List of Figures**

2.1 Task Graph 7

3.1 Project Development Approach 9

4.1 Use Case Diagram 13

4.2 Sequence Diagram 14

5.1 Graphical Representation of Event and Subscriptions 17

**CHAPTER 1**

**INTRODUCTION**

**1.1 Purpose**

The content-based publish/subscribe paradigm is a ﬂexible many-to-many communication model that meets the demands of many modern distributed applications, such as information ﬁltering, selective content dissemination, location-based services, and workload monitoring and management.

Event matching is the process of checking high volumes of events against large numbers of subscriptions and is a fundamental issue for the overall performance of a large-scale distributed publish/subscribe system.

In order to distribute the load and be scalable, a content-based publish/subscribe system often uses a network to route and forward subscriptions and events, which consists of multiple connected brokers. Compared with the centralized architecture where all subscriptions and events are sent to a single broker, the distributed system is more ﬂexible and failure-resistant. Whenever an event is received by a broker, event matching is immediately performed, returning the matched subscriptions and their routing information.

In a large-scale publish/subscribe system; it is possible that there are millions of subscriptions maintained by brokers and millions of events to be matched every second. Therefore, to improve the matching speed of brokers is of great importance to large-scale content-based publish/subscribe systems.

**1.2 Concept of Content Based publish subscribe system**

**Deﬁnition 1: Events**

Clients who publish events are called publishers. An event is an observable occurrence, which is also called message, publication or notiﬁcation in some literatures.

Usually, an event is expressed as a conjunction of attribute-value pairs. As a convention, every attribute appears only once in an event expression.

For example, {(temperature = 35),(humidity = 15)} is an event describing the weather conditions. The set of attributes appearing in the event expression is deﬁned as A ={a1,a2,...,am} and the number of attributes in the set A is denoted by m. It is assumed that the value of attributes is integer.

**Deﬁnition 2: Constraints**

A constraint is a condition speciﬁed on an attribute selected from A. Attribute is one of the attributes in A.

Value1 and value2 are bounded by the value domain of the attribute and value1 is not larger than value2. Type deﬁnes the data type of the attribute with a value domain, which can be any of {integer, double, string}.

**Deﬁnition 3: Subscriptions**

Clients who issue subscriptions are called subscribers. A subscription is an expression of subscribers’ interests in some events, which is also used to route and forward events from the publishers to the target subscribers.

Each subscription is identiﬁed by a unique subID and is speciﬁed as a conjunction of multiple range constraints. The number of range constraints contained in a subscription is not larger than m, where m is the number of attributes appearing in events.

A subscription matches an event if all the range constraints contained in the subscription are satisﬁed when they are assigned the corresponding attribute values of the event.

**Deﬁnition 4: Brokers**

Brokers are also called servers or proxies. In non-P2P computing environments, a broker is a specialized server that is responsible for routing and forwarding subscriptions and events. In P2P environments, some clients also act as brokers.

**Deﬁnition 5:Event Matching**

Event Matching Given a set of n subscriptions S = {s1,s2,...,sn} and an event e, the task of event matching is to ﬁnd all subscriptions from S which match e. The set of the matched subscriptions Sm is a subset of S, Sm⊆ S. Sm = {si | si∈ S ∩ si matches e}

**Definition 6: Publish/Subscribe Systems:**

A typical publish/subscribe system consists of subscribers, publishers, and a network of brokers. The publishers input events into the system while the subscribers submit subscriptions to the system. The core of publish/subscribe systems is to transmit events from the publishers to the target subscribers as quickly as possible

**1.3 How doesREINwork?**

REIN (REctangleINtersection), a fast event matching approach for content-based publish/subscribe systems. The key idea behind REIN is to quickly ﬁlter out unlikely matched subscriptions rather than to determine whether a subscription is matched or not by counting its satisﬁed component constraints.

It is assumed that each subscription is composed of multiple range constraints and each range constraint is a condition speciﬁed on an attribute with a low value and a high value. The range constraint is satisﬁed if an attribute value is located in the range formed by the low value and the high value.

The attributes appearing in events form a high-dimensional space. In this space, a subscription is a high-dimensional rectangle (for short rectangle) and an event is a point. Therefore, the matching problem is equivalent to the point enclosure problem. We enlarge a point (event) into a high-dimensional cube (for short cube) to transform the point enclosure problem into the rectangle intersection problem

**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 Study of Currently Existing Systems**

**i. Traditional REIN:**

Parallel REIN (REctangleINtersection), a fast event matching approach for content-based publish/subscribe systems.

The key idea behind REIN is to quickly ﬁlter out unlikely matched subscriptions rather than to determine whether a subscription is matched or not by counting its satisﬁed component constraints.

It is assumed that each subscription is composed of multiple range constraints and each range constraint is a condition speciﬁed on an attribute with a low value and a high value.

The range constraint is satisﬁed if an attribute value is located in the range formed by the low value and the high value.

Attributes appearing in events form a high-dimensional space. In this space, a subscription is a high-dimensional rectangle (for short rectangle) and an event is a point.

Therefore, the matching problem is equivalent to the point enclosure problem. We enlarge a point (event) into a high-dimensional cube (for short cube) to transform the point enclosure problem into the rectangle intersection problem.

An efﬁcient index structure is also designed to address the rectangle intersection problem by using bit operations.

**2.2 Drawbacks of Existing System**

Traditional REIN does not use any type of parallelization. The REIN uses sequential process to match the subscription with an event. The time required for the traditional REIN increases due to the use of this sequential methodology.

In order to improve matching efﬁciencies, the size of buckets is important. Given the number of subscriptions, there is a critical point for the number of buckets. After the critical point, the performance of REIN degrades with more buckets.

**2.3 Proposed System**

**i. Parallel REIN:**

Subscription Parallelism:

* Parallelism is used in case of parallel REIN for Event matching.
* More than one subscription execute at the same time on different threads.

Subscriptions:

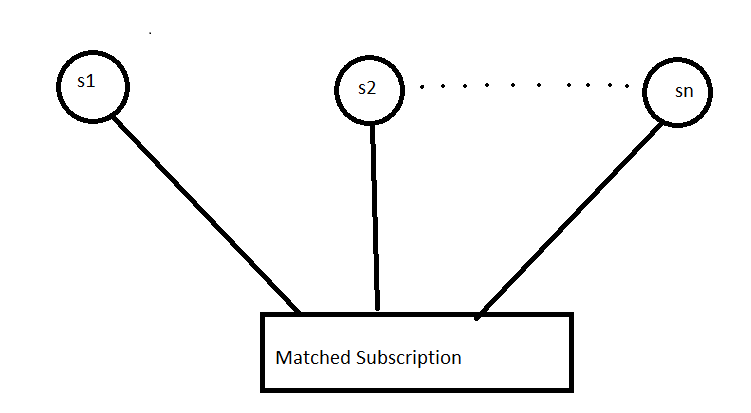
* In parallel REIN, subscriptions which are marked (1) are considered to be unmatched.
* The subscriptions which are unmarked (0) for all the attributes are the final matched subscriptions.

**Task Graph:**

Following is the task graph for Parallel REIN:

Here n= number of subscriptions.

Multiple subscriptions are matched with an event at a same time on different threads.



**Fig 2.1Task Graph**

**2.4 Advantages of Proposed System**

The most important advantage of parallel REIN is the time required time for the event matching. As Traditional REIN does not uses any type of parallelization for the event matching the time required by the traditional REIN is more.

In case of parallel REIN more than one subscription is matched with an event at the same time on different threads. This reduces the matching time. The result is combined result of all the threads. If we increase the number of threads the matching time is reduced.

**CHAPTER 3**

**PROJECT PLANNING**

Project planning is part of project management, which relates to the use of schedules such as Gantt charts to plan and subsequently report progress within the project environment. Initially, the project scope is defined and the appropriate methods for completing the project are determined. Following this step, the durations for the various tasks necessary to complete the work are listed and grouped into a work breakdown structure. The logical dependencies between tasks are defined using an activity network diagram that enables identification of the critical path. Float or slack time in the schedule can be calculated using project management software. Then the necessary resources can be estimated and costs for each activity can be allocated to each resource, giving the total project cost. At this stage, the project plan may be optimized to achieve the appropriate balance between resource usage and project duration to comply with the project objectives. Once established and agreed, the plan becomes what is known as the baseline. Progress will be measured against the baseline throughout the life of the project. Analyzing progress compared to the baseline is known as earned value management.

**3.1 Project Development Approach**

We have used Iterative and Incremental Development model (IID) for our project development. This development approach is also referred to as Iterative Waterfall Development approach. Iterative and Incremental Development is a software development process developed in response to the more traditional waterfall model.



**Fig.3.1Project Development Approach**

The basic idea behind iterative enhancement is to develop a software system incrementally, allowing the developer to take advantage of what was being learned during the development of earlier, incremental, deliverable versions of the system. Learning comes from both the development and use of the system, where possible. Key steps in the process were to start with a simple implementation of a subset of the software requirements and iteratively enhance the evolving sequence of versions until the full system is implemented.

At each iteration, the procedure itself consists of the Initialization step, the Iteration step, and the Project Control List. The initialization step creates a base version of the system. The goal for this initial implementation is to create a product to which the user can react. It should offer a sampling of the key aspects of the problem and provide a solution that is simple enough to understand and implement easily. To guide the iteration process, a project control list is created that contains a record of all tasks that need to be performed. It includes such items as new features to be implemented and areas of redesign of the existing solution. The control list is constantly being revised as a result of the analysis phase.

The iteration involves the redesign and implementation of a task from project control list, and the analysis of the current version of the system. The goal for the design and implementation of any iteration is to be simple, straightforward, and modular, supporting redesign at that stage or as a task added to the project control list. The code can, in some cases, represent the major source of documentation of the system. The analysis of iteration is based upon user feedback, and the program analysis facilities available. It involves analysis of the structure, modularity, usability, reliability, efficiency, and achievement of goals. The project control list is modified in light of the analysis results.

During the implementation of the project by this approach, a step called V&V i.e. Verification and Validation is carried out at certain intervals.

**CHAPTER 4**

**SYSTEM DESIGN**

During analysis, the focus is on what needs to be done intendment of how it is done. During design, decisions are made about how the problem will be solved, first at a high level, then at increasingly detailed levels. System design is the first stage in which the basic approach to solving the problem is selected. During system designing the overall structure and style are decided. The system architecture is the overall organization of the system into components called system. System design deals with transforming the algorithm, as described in the synopsis document, into a form that is implement able using the programming language. Certain items such as modules, relationships among identified modules, data structures, relationships between the data structures, and algorithms for implementation should be designed during this phase.

* 1. **Requirement Specification**

**4.1.1 Optimal Software Requirements**

* 1. OpenMP
  2. Operating System – Ubuntu 14.04 LTS
  3. Language – CPP

**4.1.2 OPENMP:**

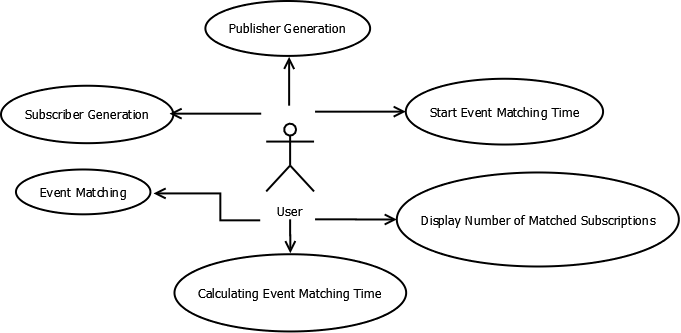
* OpenMP (Open Multi-Processing) is an application programming interface (API) that supports multi-platform memory multiprocessing programming in C, C++and Fortran on most platforms, processor architectures and operating systems.
* It consists of a set of compiler directives, library routines, and environment variables that influence run-time behavior.
* The OpenMP functions are included in a header file labeled omp.h in C/C++.
* OpenMP uses a portable, scalable model that gives programmers a simple and flexible interface for developing parallel applications.
* OpenMP is an implementation of multithreading, a method of parallelizing whereby a master *thread* (a series of instructions executed consecutively) *forks* a specified number of slave *threads* and the system divides a task among them. The threads then run concurrently, with the runtime environment allocating threads to different processors.
* The section of code that is meant to run in parallel is marked accordingly, with a preprocessor directive that will cause the threads to form before the section is executed.
* By default, each thread executes the parallelized section of code independently.
  1. **UML Diagrams**

UML is a standard modeling**language,** not a**software development process**. UML 1.4.2Specification explained that process:

1. provides guidance as to the order of a team’s activities,
2. specifies what artifacts should be developed,
3. directs the tasks of individual developers and the team as a whole, and
4. offers criteria for monitoring and measuring a project’s products and activities.

**4.2.1 Use case Diagram of a system**

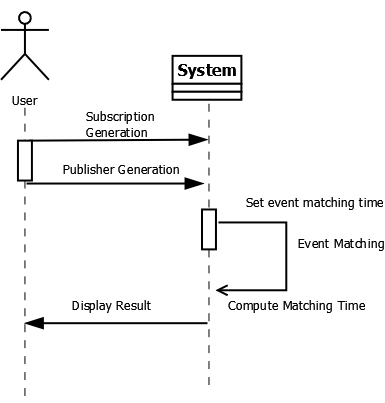
The use case diagram of a system covers all the use cases of functionality provided by a system. Use case diagram gives the overall view of what the system is going to do, how is not specified. The use case diagram drawn below is application domain oriented. This has no relation with the execution domain of the system.



**Fig.4.1: Use Case Diagram**

**4.2.2 Sequence Diagram of a system**

The sequence diagram shown below shows the timeline of the system working. It shows the lifetime of the objects specified horizontally. The messages shared between them while they are communicating with each other are shown using the arrows.



**Fig.4.2: Sequence Diagram**

**4.3 System characteristics**

* Subscriptions and Events are randomly generated.
* System gives count of matched subscriptions with event.

**4.4 Attributes**

**4.4.1 Usability:**

The user needs not to do any efforts to compute count of matched subscriptions with event. Proposed mechanism provides automation.

**4.4.2 Availability:**

System will be available for 24x7. So, the model runs at every time.

**CHAPTER 5**

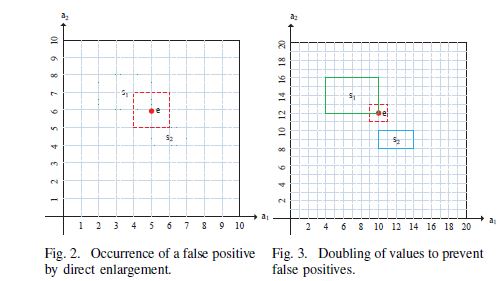
**PROJECT OVERVIEW**

**5.1 Overview**

Most existing methods compute the matched subscriptions by counting the number of satisﬁed constraints, such as SIENA and TAMA . The satisﬁed component constraints are identiﬁed ﬁrstly. Then the matched subscriptions are picked out in terms of these satisﬁed constraints. On the contrary, REIN ﬁrst marks unlikely matched subscriptions in a bit set by applying efﬁcient bit operations. The unmarked bits represent the matched subscriptions.

As given a set of subscriptions S and an event e, event matching is to ﬁnd all subscriptions from S that match the event e. The set of attributes A = {a1,a2,...,am} appearing in events forms a m-dimensional space. In the space, events are points and subscriptions are rectangles. Therefore, the event matching problem is equivalent to the point enclosure problem, which ﬁnds all rectangles that contain a given point.

We transform the point enclosure problem into the high-dimensional rectangle intersection problem. Therefore, the matched subscriptions are those rectangles that intersect a given cube which is enlarged from a point representing an event. Based on the transformation, we design an efﬁcient index structure to speed up the checking of rectangle intersection in the high-dimensional space.

.

**Fig. 5.1 Graphical Representation of Events and Subscriptions**

**5.2 Definition: REIN**

REIN is an exact matching approach for content-based publish/subscribe systems. Therefore, some techniques should be applied to prevent the occurrence of false positives. It is assumed that the values of an attribute are integer taken from a value domain which is bounded by the minimum and the maximum values. The smallest unit of integer is 1. In order to prevent false positives, the value domain of each attribute is doubled.

The doubling of value domains and values is efﬁcient to avoid false positives. Due to the effect of doubling, the values of range constraints and events are even integer. However, an enlarged cube is speciﬁed by odd integer. A rectangle either intersects or disjoints with an enlarged cube. If a rectangle r does not contain a point p, r will never intersect the cube enlarged from p, which prevents the happening of false positives. The prevention of false positives establishes a foundation of REIN.

**5.3 Index structure**

The index structure of REIN consists of multiple bucket lists. The number of bucket lists is 2m, where m is the number of attributes appearing in events. For each attribute, two bucket lists are constructed. One bucket list is for the low values of the range constraints speciﬁed on the attribute. Another is for the high values of the range constraints.

A bucket list is constructed by dividing the value domain of an attribute into cells and realizing the mapping from the cells to the buckets. All values belonging to a cell are mapped to the corresponding bucket. When a constraint value is mapped to a bucket, the constraint value and the corresponding subID are inserted into the bucket. The number of cells divided on a value domain is determined by multiple factors. One is the stability of subscriptions.

For a publish/subscribe system where the subscriptions are relatively static, less cells can be divided and the items in each bucket can be sorted on the constraint values to obtain better performance. Otherwise, more cells are needed to reduce the cost of subscription modiﬁcations. Another factor is the number of subscriptions. In order to improve matching efﬁciencies, the size of buckets is important. Given the number of subscriptions, there is a critical point for the number of buckets. After the critical point, the performance of REIN degrades with more buckets.

**5.4 Event Matching**

The procedure of event matching in REIN is straightforward. When matching events, a bit set is initialized in which the number of bits equals the number of subscriptions (line 3). All the unmatched subscriptions are marked in the bit set.

Given an event, an enlarged cube is generated which is speciﬁed by {v11,v12,v21,v22,...,vm1,vm2}, where vi1 is not larger than vi2 for all 1<= i<= m (line 4). For each attribute, we ﬁnd out all the subscriptions that

* + 1. The high value of the range constraint speciﬁed on the attribute is less than the low value of the attribute of the cube (line 6−12)
    2. The low value of the range constraint speciﬁed on the attribute is larger than the high value of the attribute of the cube (line 13−19).
    3. The unset bits in the bit set represent the matched subscriptions (line 21−25).

The event matching algorithm is shown below.

**Algorithm 1 Event Matching:**

1: input: event e

2: output: matched subIDs ID

3: initialize a bit set of size n;

4: generate a cube {v11,v12,v21,v22,...,vm1,vm2} from e;

**The following loop(5) is parallelized using openmp**

5: for (i =1to m) do

6: map vi1 to the bucket b in the bucket list for the high values of the range constraints speciﬁed on attribute i;

7: for (each item t in b) do

8: if t.val< vi1 then

9: mark the t.ID in the bit set;

10: end if

11: end for

12: mark the bits for the IDs stored in the buckets before b;

13: map vi2 to the bucket b in the bucket list for the low values of the range constraints speciﬁed on attribute i;

14: for (each item t in b) do

15: if t.val> vi2 then

16: mark the t.ID in the bit set;

17: end if

18: end for

19: mark the bits for the IDs stored in the buckets after b;

20: end for

21: for (each bit in the bit set) do

22: if (the bit is unmarked) then

23: add the corresponding subID to ID;

24: end if

25: end for

**CHAPTER 6**

**RESULTS**

**Graph 1: Subscription vs Time (microseconds)**

The following graph shows the Subscription vs Time graph. Here the range of subscription is 7000-12000.

* The above graph shows time taken by specific number of subscriptions for event matching.
* Here only 1 publisher is used and number of attributes used is 10, number of threads used is 2.

**Result Analysis:**

The graph shows that the time required by the serial REIN is more than parallel REIN. The result shows that the use of OPENMP reduces the time of execution by simultaneous execution on different threads.

**Graph 2: Attributesvs Time (microseconds)**

The following graph shows the Attributes vs Time graph. Here the range of attribute is 5-15.

* The above graph shows time taken by specific number of attributes for event matching.
* Here, only 1 publisher is used and number of subscription used is 7500, number of threads used is 2.

**Result Analysis:**

The graph shows that the time required by the serial REIN is more than parallel REIN. The result shows that the use of OPENMP reduces the time of execution by simultaneous execution on different threads.

**Graph 3: Publishervs Time (microseconds)**

The following graph shows the Publisher vs Time graph. Here the range of publisher is 1-10.

* The above graph shows time taken by specific number of publishers for event matching.
* Here, the number of subscription used is 7500 and the number of attributes used is 10.

**Result Analysis:**

The graph shows that the time required by the serial REIN is more than parallel REIN. The result shows that the use of OPENMP reduces the time of execution by simultaneous execution on different threads.

**CHAPTER 7**

**PROJECT OUTCOMES AND FUTURE SCOPE**

**7.1 Outcomes**

* The aim of the project was to parallelization of subscriptions.
* By using OPENMP we can parallelized the subscriptions.
* The results for various kinds of input are as above.
* So the parallelized system is working.

**7.2 Future Scope**

* The possibility of upgrade and enhancement of the system is huge, parallelization of attributes.
* Allowing system to reduce the event matching time.
* Allowing system to add functionalities for insertion, deletion of subscription.
* Improving range of subscriptions.
* Design of efficient index structure.

**CHAPTER 8**

**CONCLUSION**

In this paper, we present parallel REIN, a fast event matching approach for large-scale content-based publish/subscribe systems. The main idea behind parallel REIN is to quickly ﬁlter out unlikely matched subscriptions rather than to identify matched subscriptions by counting satisﬁed component constraints.Experimental results show that parallel REIN outperforms its counterparts to a large degree, especially in the case where the selectivity of subscriptions is high and thenumber of subscriptions is large.

**BIBLIOGRAPHY**

[1]ShiyouQian,Jian Cao, Yanmin Zhu, Minglu Li, “REIN: A Fast Event Matching Approach for Content-based Publish/Subscribe Systems”, Shanghai Jiao Tong UniversityIEEE INFOCOM 2014 - IEEE Conference on Computer Communications

[2]P. Eugster, P. Felber, R. Guerraoui, and A. Kermarrec, “The many faces of publish/subscribe,” Computing Surveys (CSUR), vol. 35, no. 2, pp. 114–131, 2003.

[3] F. Fabret, H. Jacobsen, F. Llirbat, J. Pereira, K. Ross, and D. Shasha, “Filtering algorithms and implementation for very fast publish/subscribe systems,” in SIGMOD. ACM, 2001, pp. 115–126.

[4] A. Carzaniga and A. Wolf, “Forwarding in a contentbased network,” in SIGCOMM. ACM, 2003, pp. 163– 174.

[5] H. Jafarpour, S. Mehrotra, N. Venkatasubramanian, and M. Montanari, “Mics: an efﬁcient content space representation model for publish/subscribe systems,” in DEBS. ACM, 2009, p. 7.