Innovating R Tree to Create Summary Filter for Message Forwarding Technique in Service-Based Routing

Nguyen Thanh Long[1], Nguyen Duc Thuy[2], Pham Huy Hoang[3], and Tran Dinh Chien[1]

[1] Informatic Center of Ha noi Telecommunications
75 Dinh Tien Hoang, Hoan Kiem, Ha Noi, Viet Nam
{longptpm,chientd}@vnpt-hanoi.com.vn
[2] Research Institute of Posts and Telecommunications
122 Hoang Quoc Viet, Nghia Tan, Cau Giay, Hanoi, Viet Nam
thuynd@ptit.edu.vn
[3] Hanoi University of Science and Technology
1 Dai Co Viet Road, Hanoi, Viet Nam
hoangph@soict.hut.edu.vn

Abstract. In service-oriented routing [5], the problem for storing routing table of filters includes search predicates received from subscribers through subscription messages is an important job. When a content request happens, subscriber will create one subscription message, the subscription message stores several kinds of information, the most important content is filter that is a conjunction of some constraints [5]. One filter is denoted by F character, that has mathematical formula: $F=P_1 \Lambda P_2 \Lambda \ldots \Lambda P_n$ (1), in which P_1 is service request, has format: P_1 = 'Service_name = requested_service_name'. Every P_i (in which $i=\overline{2..n}$) is a constraint that is formed by three components: (Key, op, Value), Key is a keyword for searching, op is an operator, Value is searching condition. Key belongs to the set of name of properties of content that the requested service supplies. Op is operator that depends on the type of data of the Key. Therefore the routing table of the service based routing is a set of filters which are received from all subscribers on networks. The algorithms for inserting, updating, deleting and finding filters that match content messages have been published by service providers are very important. In this paper, We mention the technique on filter summary for storing and searching filter quickly that based on some previous researches and cluster routing [6] based on root's summary filter.

Keywords: Service based routing, filter, constraint, service, routing, tree, split, summary, R, cluster, head.

1 Overview Summary Filter Technique

Considering the overarching concept, assume that there are two filters F_1 and F_2 are requirements of one service that arising from one subscriber. We suppose that F_1 covers F_2 by the notation: $F_1 \ni F_2$ (2), if all messages satisfy conditions of F_2 then they satisfy F_1.

H. Qian and K. Kang (Eds.): WICON 2013, LNICST 121, pp. 178–188, 2013.

Symbol by: $F_1=P_{12} \Lambda P_{13} \Lambda \dots \Lambda P_{1n}$, $F_2=P_{22} \Lambda P_{23} \Lambda \dots \Lambda P_{2m}$ (3).

In that: $P_{12}=K_{12}op_{12}V_{12}$, $P_{13}=K_{13}op_{13}V_{13}$, …, $P_{1n}=K_{1n}op_{1n}V_{1n}$,

$P_{22}=K_{22}op_{22}V_{22}$, $P_{23}=K_{23}op_{23}V_{23}$, …, $P_{2m}=K_{2m}op_{2m}V_{2m}$.

For (2) we must have: $\{K_{12}, K_{13}, \dots, K_{1n}\} \subseteq \{K_{22}, K_{23}, \dots, K_{2m}\}$.

Suppose the predicate: P = K op V, called X is the value domain of the constraint, that means the values V are belonged to X that will satisfy the constraint.

If (2) is true, at the same time with conditions: $n \leq m$ and $K_{12} \equiv K_{22}$, $K_{13} \equiv K_{23}$, …, $K_{1n} \equiv K_{2n}$, and $op_{12} \equiv op_{22}$, $op_{13} \equiv op_{23}$, …, $op_{1n} \equiv op_{2n}$:

Must have: $X_{12} \ni X_{22}$, $X_{13} \ni X_{23}$, …, $X_{1n} \ni X_{2n}$.

That means all the keysand operators are contained in F_1 that are also contained in F_2, domain value of each of constraintof F_1coversdomain value of each constraintof F_2.

We suppose that F_1 and F_2 overlap each other if the following terms are satisfied:

$$X_{12} \cap X_{22} \not\equiv \emptyset, X_{13} \cap X_{23} \not\equiv \emptyset, \dots, X_{1n} \cap X_{2n} \not\equiv \emptyset \quad (5).$$

Assume specific domain of each constrainthasmagnitude of $|X|$, so that specific domain of the filter has magnitude of $\prod_{i=1}^{n} |X_i|$ (6).

When receiving a new subscription message from network interface (I) which requests service S, it has a filter is denoted by formula: $F=P_{12} \Lambda P_{13} \Lambda \dots \Lambda P_{1n}$. We have to check whether F is covered by an existing filter. If there is no filter, have to check whether it has any common constraints with existing filter emitted from an node interface I. For doing this task efficiently We have to apply and innovate R tree, in this paper it is called R^+, this tree nowadays is used in many different fields. For example R tree is applied for storing space objects in Google MAP, digital Map, System Paging file with high efficiently in storing and searching.

2 Improving R^+Tree to Store and Search Filter

2.1 Structure of R^+ Tree

Structure of leaf node: it is a set S that consists of n elements, each element consists of two items P and FS: P is a pointer that points to a filter F that is indexed in this R^+ tree and FS is a filter summary that covers filter F: FS ∋ F.

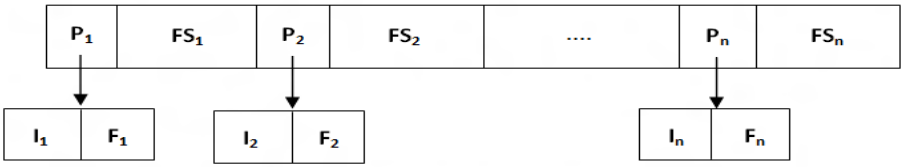

Fig. 1. The structure of a leaf node of R^+ tree

The filters are added by the algorithm that will be presented in the following section.

a) The leaf node stores the set of summary filters of some filters in routing table that are indexed in the R^+ tree that are received from routers on networks.

b) Each filter is indexed in R^+ tree consists of two components: I and F, in which I is address of router that have sent subscription message, F is a filter including of some constraints.

c) The inner nodes of tree that are not the leaf nodes, each node stores a set of summary filters of its child nodes. Similarly leaf node, the structure of an inner node is as follows:

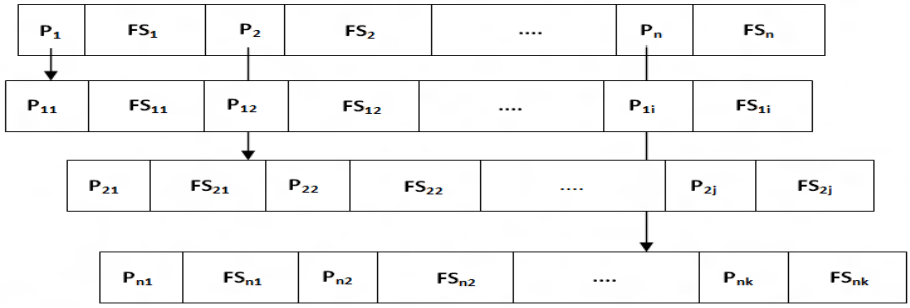

Fig. 2. The structure of an inner node in R^+ tree

d) Thus, each inner node stores a set of n elements in which each element consists of two components: P and FS, P is a pointer that points to an inner node or leaf node of the R^+ Tree. FS is a summary filter that covers the whole set of n summary filters of the node pointed by P. This means: $FS=\bigcup_{i=1}^{n} FS_i$. The following diagram describes the covering relationship with n=6:

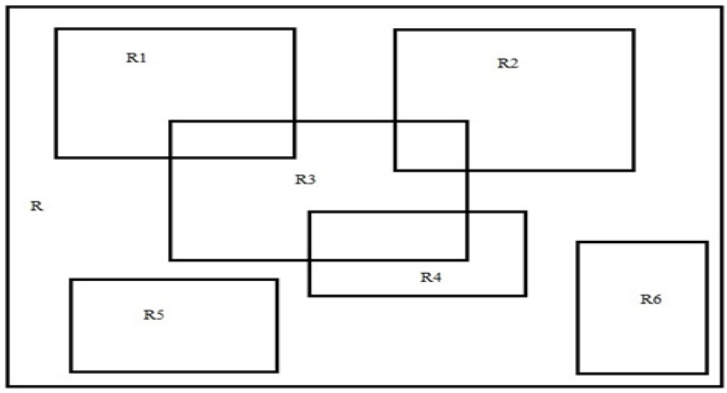

Fig. 3. The covering model with R filter covers its six child filters R_1 to R_6

Thus We have:

a) The Root node stores a set of the highest level summary filters of its child nodes that form the routing table. Therefore the summary filter of root node covers the whole routing table. So that when checking if a filter belongs to routing table, at first

examine whether it is covered by summary filter of the root node. If it is not covered, Weconclude this filter doesn't belong to the routing table.

b) Each leaf node or inner node has anumber of child nodes which is bound by a lower bound m and a upper bound M: $\text{Amount}_{\text{child}} \in [m, n]$, m and M∈ N. Usually We choose m=$\left\lfloor \frac{M}{2} \right\rfloor$ or m=$\left\lfloor \frac{M}{3} \right\rfloor$ because when number of child nodes of one node reaches M+1, We have to split this node into two nodes so that each node has a approximately equal number of child nodes.

c) R^+ tree is a balance tree, this means that the height from any leaf node to root node of tree is equal. The proving is based on building tree with new filter is only added to some leaf node. At the beginning, the tree has only one node which is both root and leaf node simultaneously. When number of filters are added to node is greater than M, createa new leaf node, split its child nodes into 2 leaf nodes. Then create anew root node, it is parent of these two leaf nodes. Assume that at some time, all inner nodes and leaf nodes and root are having M child nodes. The problem is to insert one filter to some leaf node (L), the solution is to create a new leaf node (L_1), and split the set of filters of current node into two subsets of leaf nodes of L and L_1. At this time parent node (N) of current leaf node L has (M+1) child nodes, so have to create new node N_1 with the same level of N, split child nodes of N into two nodes N and N_1. Gradually applying the same procedure getting to the root node (R), at root node, create new node (R_1), divide the set of child nodes of root node into two nodes R and R_1. Create new root node R' points to R and R_1. At this moment the height of all leaf nodes is increased by one. So that R^+ tree is balance tree.

d) Thus the height of the tree from aleaf node to the root node is satisfied the following un-equation system: $\log_M(N) - 1 \leq \text{Height}_{R^+} \leq \log_m(N)$, in which N is number of nodes of tree. Prove: i) At level 0, We have one node that is root node (equal to M^0), at level 1, We have maximum number of M^1 nodes, minimum number of m^1 nodes,..., at level n, We have maximum number of M^n nodes and minimum number of m^n nodes. *ii)* Consequently, with R^+ tree with high is n We have nodes of n+1 levels: 0, 1, 2, ..., n, total of nodes R^+ tree is N which has to satisfy un-equation system: $N \leq \sum_{i=0}^{n} M^i = \frac{M^{n+1}-1}{M-1}$ and $N \geq \sum_{i=0}^{n} m^i = \frac{m^{n+1}-1}{m-1}$ => $m^n \leq N \leq M^{n+1}$ => $[\log_M(N) - 1] \leq n \leq \log_m(N)$.

2.2 Establishing and Searching Algorithms

2.2.1 The Establishing Algorithm

At first R^+ tree has only one node, this is both leaf and root of tree, this node points to a filter is created to store the filter of new subscription message that has been received at current router. When adding new filter F to R^+ tree, We have to find a leaf node N, add F to N. The process for finding node N begins from root node R:

a) If R has no child node then: N=R;

b) If R has some child nodes, that means R^+ tree has some inner nodes, assume that R consists of n components: C_1, C_2, ..., C_n, denote by S = {C_1, C_2, ..., C_n}. In which each C_i has two items: P_i and FS_i, P_i points to its child node and FS_i is summary filter of its child node filters.

For each component C_i in S: *i)* Find minimum filter F_{min} satisfies: $(FS_i \cup F_{min}) \ni F$, it means FS_i is extended by minimum filter to cover filter F. *ii)* After this loop finished We will find out $F_{min_{final}} = \min\{F_{min_i}\}$ it is respective to C_{min}. *iii)* If there are many C_i satisfy this condition, choose the component that has $|FS_i|$ is minimum.

c) Assume finding out C_{final} that has pointer points to its child node R_{child}. Assign temporary node $N_{temp} = R_{child}$.

d) Apply the step (b) for N_{temp} continuously until N_{temp} is leaf node.

e) Assign: $N = N_{temp}$.

f) Top up the set of filters that are stored by N with F.

2.2.2 Searching Algorithm

a) The searching algorithm finds a leaf node N that has a filter that conforms most to a given filter F. Starting from the root of tree, assume that R has n components: $S=\{C_1, C_2, ..., C_n\}$. Assign R to $N_{current}$ and execute step (b).

b) Execute for each component C_i of $N_{current}$: C_i consists of two items P_i and FS_i, P_i points to $N_{current}$'s i^{th} child node denoted by $Child_i$, FS_i is summary filter of all filters that are stored by $Child_i$.

Check condition FS_i and F have common part, if it's true then continue to check if $Child_i$ is leaf node then execute (c) else assign $Child_i$ to $N_{current}$ and execute (b) recursively;

c) We calculate $FS_{common}$is common part of FS of $Child_i$ and F. Denote S_F is a set of some FS_{common} found. Add FS_{common} to S_F. Go to (b) for next component of $N_{current}$.

d) Find some maximum items (FS) from S_F, that belong to some leaf nodes $\{N_{max}\}$. Thus the set $\{N_{max}\}$ is result of algorithm.

2.2.3 Algorithm for Adding a Filter to R^+ Tree

This algorithm adds a new filter F to a R^+ tree: First, establish R^+ tree if it does not exist. Secondly,check if this filter exists in this R^+ tree. If it's true then exit the algorithm, otherwise find a leaf node in tree to store this filter. In this section introduce the algorithm to insert a filter to a leaf node:

a) Create a filter node to store filterF and I, I is interface address of the route has emitted the subscription message which containedthe filter F, *i)* create a filter node FN, *ii)* add filter F to FN and assign interface address I to FN: FN.filter=F; FN.Interface=I;

b) Assume current node is N, if number of components of N is lower than M then *i)* create new component C_{new} of N, it has two elements P and FS; *ii)* Calculate filter summary of F and assign to FS, P points to FN: N.FS=Summary(F); N.P=FN. Otherwise go to step (c).

c) Have to do *i)* create new node N_1, *ii)* split the set of all components of N in addition C_{new} into two subsets fairly, each of which contains components of one node N or N_1. Check the condition, whether N has no parent node, if it's true new node $P_N$iscreated, this node has two components C_1 and C_2. Assign $C_1.FS = Summary(N_1)$, $C_1.P = N_1$, $C_2.FS = Summary(N_2)$ and $C_2.P = N_2$. Now P_N is the root of R^+ tree, finish algorithm (the root of tree has at least two children). Otherwise, go to step (d).

d) N has a parent node, this node is denoted by P_N, P_N stores component C_N, C_N has two elements FS and P, P points to N, *i)* if N_1 hasn't been created in (c) then

recalculate filter summary for N, assign to FS then go to step (f), *ii)* otherwise go to step (e).

e) Create a new component C_{new} of P_N, C_{new} has two elements P, FS, *i)* calculate filter summary of N_1, assign to FS: C_{new}.FS=Summary(N_1); *ii)* assign P to N_1: C_{new}.P=N_1. Go to step (g).

f) Check filter summary of P_N, if it has changed, have to assign N=P_N, then go to step(d). Continue until N is root node.

g) Check number of components of P_N, if this number more than M, assign N=P_N, go to step (c). Continue until go to the root node then finish algorithm.

2.2.4 Algorithm for Deleting a Filter from R^+ Tree

Assume need to remove a filter F from R^+ tree. First, leaf node N has been found which is leaf node and may store F. Secondly check whether N really stores F. The purpose of this algorithm is to find N by some following steps:

a) If R^+ tree have only one node R this is both leaf and root of tree then: N=R;

b) Denote N_{temp} is current processing node: N_{temp}=R; N_{temp} has n components: {C_1, C_2, ..., C_n}, for each component C_i of N_{temp} has two elements P_i and FS_i:

c) If FS_i and F have common part then assign N_{temp}=C_i.P_i, if N_{temp} isn't leaf node then continue step (b); otherwise: N= N_{temp}, assume N stores a set of n filters {F_i}, check each filter $F_i$of Nto see whether F_i and F are equal, *i)* if this condition is true then remove F_i from N, recalculate R^+ by following regulating algorithm, this algorithm is finished; *ii)* if no filter satisfies then continue step (c) for next component of N_{temp}.

d) Continue until F is found or all tree's nodes are scanned.

2.2.5 Algorithm for Regulating R^+ Tree

a) Starting from the leafnode S from which a filter is removed.

b) If the number of components of S isn't satisfied the condition (3) then remove S from R^+, add S to a set Q which stores temporarilyall nodes have removed of R^+ and go to step (c). If it is satisfied then finish algorithm.

c) If S has parent node then assign S=S.parent, *i)* if S is not root of the tree then go to step (b); *ii)* otherwiseadd root of the tree to Q.

d) Get last node S has just been put to set Q, scan this subtreefor adding its filters to original R^+ tree.

3 Algorithm for Splitting One Node of R^+Tree into Two Nodes

3.1 Processing Steps of the Algorithm

When number of components of one node is more than upper bound M, thus having to split this node into two nodes with each node has about $\left\lfloor \frac{M+1}{2} \right\rfloor$ components. Assume current processing node is N:

a) Create new node N_1, get $\left\lfloor \frac{M+1}{2} \right\rfloor$components from N to N_1 by one of two algorithms that are described morespecific below.

b) *i)* If parent of N exists then add N_1 to the set of children of N and continue step (c); *ii)* otherwise create new node that will be new root of tree R. Add N and N_1 to the set of children of R, finish the algorithm.

c) Assign N=N.Parent, *i)* if number of child nodes of N is more than M then go to step (a); *ii)* otherwise finish the algorithm.

3.2 Algorithm Evaluation

The algorithm has maximum complexity in case of regulating tree from one leaf node to the root node of the tree. This is $O[\log_m(N)*K]$, in which N is number nodes of the tree, K is complexity of the splitting algorithm.

4 Algorithm for Splitting Components of a Node into Two Subsets

4.1 Quadratic Algorithm

4.1.1 Algorithm Specification

Assume S is a set of summary filters of current processing node that needs to be splited: $S=\{F_1, F_2, ..., F_n\}$ satisfies equation: n=M+1. The requirement of algorithm is: splitting S into two subsets S_1 and S_2 with almost the same size. Require to choose two beginning elements for these two subsetsby the following algorithm:

Algorithm 1:

a) Give filter $F_{temp}$is temporary filter, id_1, id_2 are two integers for storing two indexes of choosen filters, assign filter $F_{temp} = \emptyset$, $id_1=-1$, $id_2=-1$;

b) For each filter F_i in theset S of filters execute (c);

c) For each filter F_j in the set S execute (d);

d) Calculate filter F_{ij} that covers both filter F_i and F_j; calculate: $F'_i = F_{ij} \setminus F_i$, $F'_j = F_{ij} \setminus F_j$, if $|F'_i \setminus F'_j| > |F_{temp}|$ (in which \ is filter subtract operator) then assign: $id_1=i$, $id_2=j$ and $F_{temp}= |F'_i \setminus F'_j|$.

When this algorithm finished, gain id_1 and id_2 are indexes of two filters (FI_1 and FI_2) for beginning filters of two sets S_1 and S_2.

Choose next elements for each setS_i and $S_j$according to following algorithm:

Algorithm 2:

a)Assume$F_1$and$F_2$are summary filters of S_1 and S_2respectively, assign $F_1=\emptyset$, $F_2=\emptyset$.

b) For each filter $F_i$of the set S except FI_1 and FI_2 execute (c).

c)Calculate$F_1'=(F_1 UF_i)\setminus F_i$, $F_2'=(F_2 UF_i)\setminus F_i$,in which denote $(F_1 UF_2)$is filter that is result of extending$F_1$to cover F_2. Go to step (d).

d) If $F_1' > F_2'$then put F_i into S_2 and assign $F_2=F_2 UF_i$ (4).

e) If $F_1' < F_2'$ then put F_i into S_1 and assign $F_1=F_1 UF_i$ (5).

g) If $F_1'= F_2'$ then put F_i into what set depends on the set that has number of elements isless than other set, if two sets have equal number of elements then put $F_i$into a random set, recalculate (4) or (5) respectively. Go to (b) for the next item of S.

4.1.2. Evaluating Algorithm

The algorithm 1 has complexity of $O(n^2)$, the algorithm 2 has complexity of O(n). Therefore the complexity of Quadratic algorithm is $O(n^2)$.

4.2 Linear Algorithm

4.2.1. Algorithm Specification

We have to split the set S of summary filters of the current processing node into two approximate equal sets S_1 and S_2.

Some assumptions:

a)Each filter of S has a set of predicates, each predicate has a value domain D.

b) Each value domainDhas lower boundD_{min} and upper boundD_{max}, so that D=[D_{min}, D_{max}].

The algorithm for finding two first elements of two subsets S_1 and S_2

a) For each key K, find maximum lower bound L_{max} and minimum upper bound $U_{min}$of all predicates of Swhich have key K. These bounds are belonged to two filters FL_{max}, FU_{min} respectively.

b) For each K, also find the minimum lower bound L_{min} and maximum upper bound U_{max} of all predicates of S that have key K. These bounds are belonged to two filters FL_{min} and FU_{max} respectively.

c) Calculate for each key: $V_m = \frac{U_{min} - L_{max}}{U_{max} - L_{min}}$ (6).

d) Choose two first elements of two subsets S_1 and S_2 froma pair of filters (FL_{min}, FU_{max}) which has V_m that is maximum on all their keys.

e) With (n-2) remaining filters, put each filter to one of two sets S_1 and S_2 in turn.

4.2.2 Evaluating Algorithm

The complexity of algorithm to choose two first filters of two sets S_1 and S_2 is rated by formula (|F|*|S|), |F| is number of predicates of filter F, |S| is number of filters of S.

The complexity of algorithm to put (n-2) remaining filters to two sets S_1 and S_2 is O(n), n is number of filters of the set of filters S.

5 Cluster Routing Based on Filter Summary [6]

5.1 Cluster Routing Concept

Each node builds it's own R^+ tree when it receives subscription filter from network. The root of each tree will store filter summary of it's routing table. Define a point (P) is centroid of the rectangle that is established from each filter summary. To cluster network based on centroid of subscription filter, assume that: i) the space of all filter is S, ii) S consists of some sub-spaces {S_1, S_2, ..., S_n}. So when a node on a cluster receives one filter, it will forward this filter to appropriate cluster head based on point P of this filter.

5.2 Cluster Routing Performance

Use this cluster routing that will reduce time required to find matched subscriptions with each received content message. When a node receives a content message, this content message is a point on space S. It checks to find sub-space S_i that P belongs to and S_i has cluster head (CH_i) respectively. Then the node forwards this content message to CH_i. In CH_i all matched subscriptions will be found using above search algorithm. This cluster algorithm has complexity is O(n*m), n is number of sub-spaces and m is number of predcates of each filter, it is nearly a constant.

6 Evaluating Results and Future Development

6.1 The Results of Executing above Mentioned Algorithms

a) Build a R^+ tree from a set of filters entered from keyboard or read from disk file.

All classes and modules to simulate presented algorithms have been written by C#.NET. Besides there are some algorithms for following purposes:

*i,*Check if one filter covers or overlaps other filter.

*ii,*Extend one filter to cover other filter.

*iii,*Calculate subtractionof two filters.

b) Build R^+ with number of filters changes from 100000 to 290000, measure the time in ms required to execute for each number of filters to draw the diagram based on these results as below:

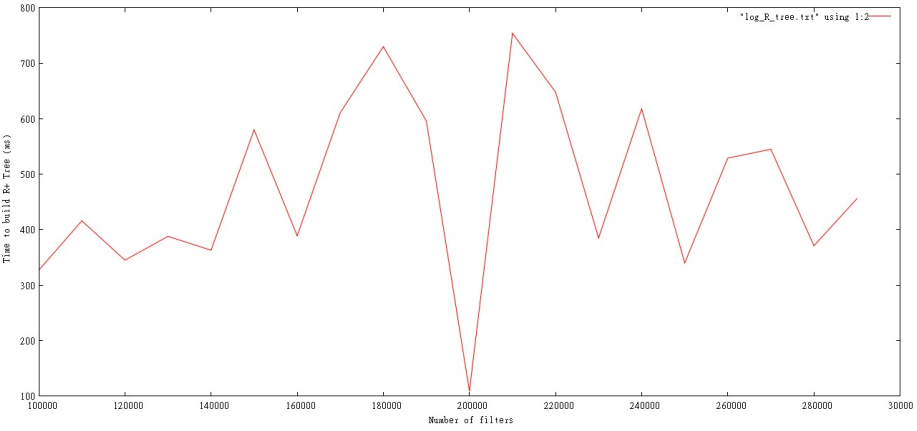

Fig. 4. Execution time in ms to build R^+ tree

As the diagram shows that the time required to build R^+ does not increase continuously when number of filters increases.

c) Build a R^+tree and use theQuadratic algorithm for splitting node with the number of filters changes from 100000 to 290000 and delete filters, measure the time required in mili second to delete filters and regulate the tree to draw diagram based on the measured results:

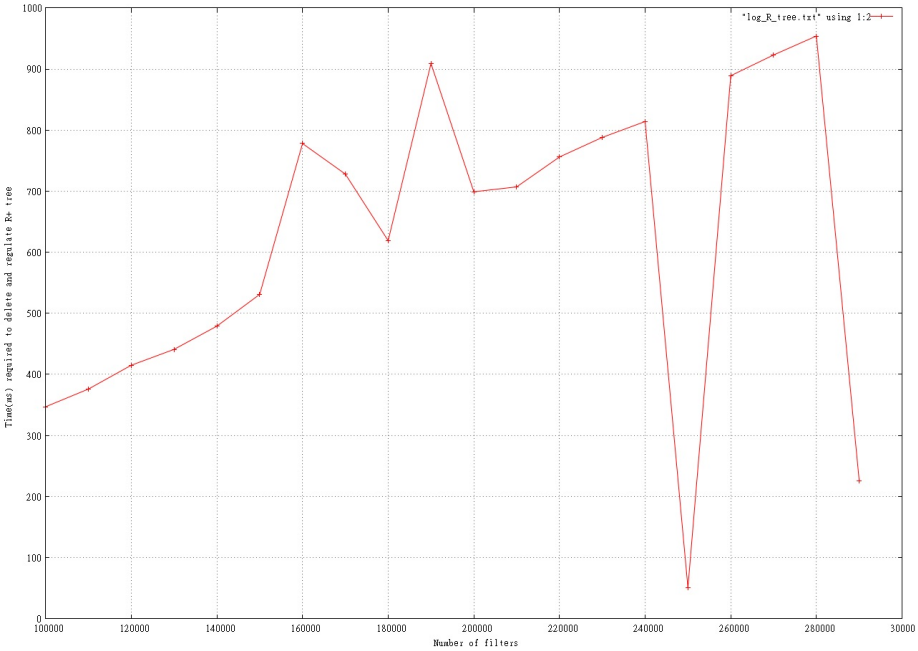

Fig. 5. Execution time in ms to delete filters and regulate R^+ tree

As the diagram shows that the time required to delete filters and regulate R^+ does not increase continuously when number of filters increases.

d) Search a filter from R^+ tree, measure searching time when number of filters is gradually changed. See following diagram, conclude that when number of filters is increased, required time is reduced.

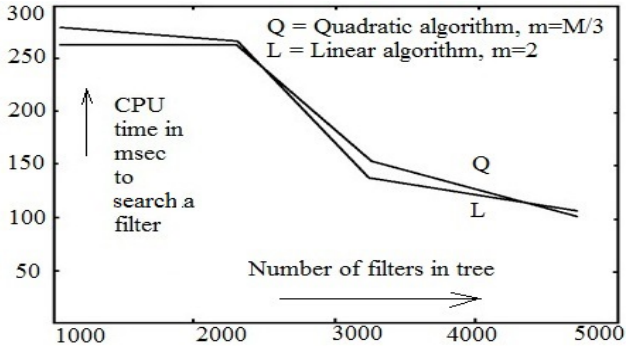

Fig. 6. Compare execution time to search filter with number of filters changed

e) Update, delete filters from R^+ tree, with concurrently regulating tree to satisfy the constraint on number of child nodes of each inner node. Draw diagram to evaluate operational time based on two splitting node algorithms and variable number of nodes of R^+ tree[2].

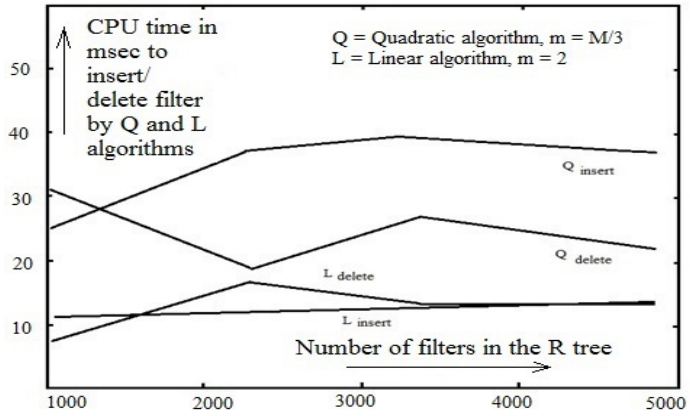

Fig. 7. Compare operational time to insert/ delete filters with change in number of filters

See this figure to get results that the time to execute is stable when number of filters increases for two algorithms to split node Q and L.

6.2 Future Developing

At the current time above algorithms have been simulated to get above operational results are on numeric value type of predicate, in the future applying above algorithms for value type of string.

Acknowledgements. I would like to thank the dedicated instructors, teachers for help and support. My office, family, brothers and friends have made many favorable conditions for studying and researchingto complete this research paper.

References

[1] Robmson, J.T.: The K-D-B Tree A Search Structure for Large Multldimenslonal Dynarmc Indexes. In: 4CM-SIGMOD Conference Proc., April 10-18 (1981)
[2] Guttman, A.: R-Trees - A Dynamic Index Structure for Spatial Searching, University of California Berkeley
[3] Guttman, A., Stonebraker, M.: Using a Relational Database Management System for Computer Added Design Data. IEEE Database Engineering 5(2) (June 1982)
[4] Yuval, G.: Finding Near Neighbors in k-dimensional Space. Inf. Proc. Lett. 3(4), 113–114 (1975)
[5] Long, N.T., Thuy, N.D., Hoang, P.H.: Research on Innovating, Evaluating and Applying Multicast Routing Technique for Routing messages in Service-oriented Routing. In: Vinh, P.C., Hung, N.M., Tung, N.T., Suzuki, J. (eds.) ICCASA 2012. LNICST, vol. 109, pp. 212–228. Springer, Heidelberg (2013)
[6] Wang, Y.-M., Qiu, L., Verbowski, C., Achlioptas, D., Das, G., Larson, P.: Summary-based Routing for Content-based Event Distribution Networks, Microsoft Research, Redmond, WA, USA (2004)