Scalable Event Matching for Overlapping Subscriptions in Pub/Sub Systems

Zhen Liu, Srinivasan Parthasarathy, Anand Ranganathan, Hao Yang

Presented by Vamsidhar Thummala
5th Feb 2008

Instructor: Prof. Jun Yang

Preliminaries

- What is a publish/subscribe system?

Applications
- Stock information delivery
- Auction system
- Air traffic control
- etc.

Pub/Sub systems are of two types
- Topic-based
- Content-based

Selectivity
- the probability that an event satisfies a given predicate

Popularity
- the number of subscriptions containing the given predicate

Problem

- Current content-based publish/subscribe systems has to deal with huge number of subscriptions
- Scalability?
- Optimal subscription evaluation is NP-Hard

Solution
- Approximation algorithm by exploiting the overlap in the subscriptions

Example

- Monitoring metropolitan area using UAV’s, subscriptions can be of type:
  - Naive Approach: Evaluate each subscription one by one

Observations

- Users often share common interests
  - subscriptions share common predicates
  - reuse the evaluated predicates
- A notification has to be sent only if
  - all the predicates in the subscription satisfies the event
- Else all the subscriptions containing the not satisfied predicate can be discarded
Analysis of problem

• Theorem:
  – Subscription evaluation problem is NP-Hard (NP-Complete?)

• Proof:
  – Reduce subscription evaluation problem to set-cover problem

Sequential Algorithm

• Returns the set of subscriptions whose predicates are all satisfied individually.
  – Perform variable bindings and value checks to determine whether the subscription is completely satisfied

• Running time
  – at-most d*optimal
  – d is the maximal number of predicates across all subscriptions

Sub-Order Algorithm - Preliminaries

• Linked chains structure

Sub-Order Algorithm - Preliminaries

• Consider an equivalence link between nodes \( n_{1,a} \) and \( n_{2,b} \), where \( 1 \leq a \leq n \) and \( 1 \leq b \leq m \)

• Cost of evaluating \( s_1 \) first and then \( s_2 \) is

\[
C = \left( \text{Cost of evaluating all nodes in } n_{1,a} \right) + \left( \text{Probability of nodes } n_{1,1}, n_{1,2}, \ldots, n_{1,n} \text{ all having solutions} \right) + \left( \text{Probability of node } n_{2,b}, \text{ having solutions} \right) \times \left( \text{Cost of evaluating } s_2 \text{ without having to evaluate node } n_{2,b} \right) + \left( \text{Probability of at least one of the nodes } n_{1,1}, n_{1,2}, \ldots, n_{1,n} \text{ not having a solution} \right) \times \left( \text{Cost of evaluating } s_2 \text{ including the evaluation of node } n_{2,b} \right)
\]
Sub-Order Algorithm - Preliminaries

- Cost of evaluating \( s_1 \) before \( s_2 \)
  \[
  C_1 = \left( \prod_{i=1}^{n} p_{s_1} \right) \left( \prod_{i=1}^{n} p_{s_2} \right) + \left( \sum_{i=1}^{n} p_{s_1} \right) \left( \sum_{i=1}^{n} p_{s_2} \right)
  \]
  
- Cost of evaluating \( s_2 \) before \( s_1 \)
- Cost difference
  \[
  C_2 = \left( 1 - \prod_{i=1}^{n} p_{s_1} \right) \left( \prod_{i=1}^{n} p_{s_2} \right) + \left( 1 - \sum_{i=1}^{n} p_{s_1} \right) \left( \sum_{i=1}^{n} p_{s_2} \right)
  \]

Sub-Order Algorithm

- Arrange the predicates in the decreasing order of selectivity among each subscription chain
- Calculate the cost differences for each pair of subscriptions (Now, the problem is TSP for which has a \( k \)-approximation solution)
- Use Kruskal’s algorithm (greedy) to form a minimum spanning tree
  - The weight of the edge \((u,v)\) is the cost savings of evaluating \( u \) before \( v \)
- Evaluate the nodes in the order given by Kruskals

Assumptions

- Selectivity of predicates is conditionally independent of selectivity of other predicate
- All predicates have the same evaluation cost
- If predicates have different evaluation cost, then approximation guarantee increases from \( d \) to \( d \times C_{max}/C_{min} \).

Experiments

- Synthetic workload

<table>
<thead>
<tr>
<th>parameter</th>
<th>description</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N_s )</td>
<td>number of subscriptions</td>
<td>2K - 8K</td>
</tr>
<tr>
<td>( N_p )</td>
<td>number of predicates</td>
<td>100 - 200</td>
</tr>
<tr>
<td>( mod )</td>
<td>popularity distribution of the predicate</td>
<td>Uniform or Zip</td>
</tr>
<tr>
<td>( min_p )</td>
<td>minimum number of predicates in a subscription</td>
<td>3</td>
</tr>
<tr>
<td>( max_p )</td>
<td>maximum number of predicates in a subscription</td>
<td>11</td>
</tr>
<tr>
<td>( N_r )</td>
<td>number of requests</td>
<td>1000</td>
</tr>
<tr>
<td>( sub_ratio )</td>
<td>ratio of subscribed subscriptions among all subscriptions</td>
<td>0.2%</td>
</tr>
</tbody>
</table>

Results

- Claim: The performance of Sequential is not sensitive to the subscription length

Points of discussion

- Only selectivity is considered
  - How to order both on selectivity as well as popularity?
- Synthetic data set is used
- Metric used: Average number of predicates
- Are the results convincing?
  - There is hardly a noticeable difference between Sequential and Sub-Order algorithm from the graphs
Points of discussion

• Unrealistic assumptions
  – Independent assumption on selectivity of predicates
• Cost model
• Prim’s for MST?
• The current model provides sort of middle layer assuming that all the predicates are retrieved (from multi-media objects) and available. Can’t we achieve better selectivity by pushing the selections down to multi-media objects?