Event Matching in Content-based Publish/Subscribe Systems

Milenko Petrovic

Outline

- Motivation
- Problem Definition
- Solutions
- Evaluation
- Conclusions

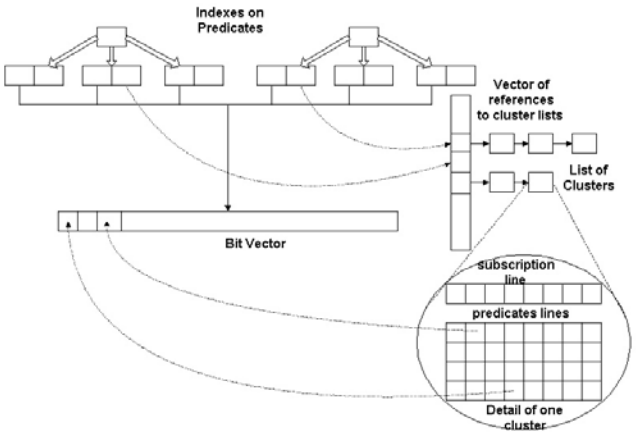
Motivation

- Web is big, time is short ("expensive")
- How to get relevant information *quickly*?
- Content-based Publish/Subscribe Systems
  - Multiple event schemas
  - Can do "query" on event *content*
  - Can query future data
- IBM P/S System @ Wimbledon 2002
  - Real-time stats to 230,000 Internet users
  - Does not scale to millions of users or events

Specifically…

- Given an event e and a set of subscriptions S determine all subscriptions in S that are matched by e.
- Solution classification:
  - Tree-based: Gryphon, Gough et. al.
  - Predicate-based: counting, Hanson et. al., propagation, Fabret et. al.
- Best results so far – Fabret et. al.

Event Matching



Creating Clusters

- When: offline (static), online (dynamic)
- How: quantify the cost of matching
  - Time = retrieving the indexes + hashing cost for relevant hash tables + checking the relevant clusters
  - Space = hashing structures + subscription clusters

Offline Algorithm

- Greedy algorithm
- Initial set of clusters = subscriptions grouped by longest common conjunction of equality predicates
- Select clusters from the initial set that minimize space + time cost
- End when cannot select any more clusters

But, …

- Event skew can make a cluster configuration suboptimal
- Solution: dynamic clustering => need to quantify how insertions, deletions and event skew affect the performance
  - Need thresholds for creating a new cluster and deleting an exiting cluster
  - Insertion and event skew cost = how close to optimal is current configuration wrt events and new subscriptions + benefit of any new clusters
  - Deletions = is the benefit of the cluster below the threshold?

Online Algorithm

- for each new event or subscription, redistribute subscriptions from clusters that have selectivity above some threshold
- while redistributing consider creating new clusters that minimize space + time cost
- delete any clusters with benefit below some threshold

Evaluation

- There is an implementation
- Comparative performance study of predicate matching algorithms
- Scenarios
  - scalability and throughout
  - adaptability to event and subscription skew
- Results confirm that the dynamic algorithm has the best performance

Conclusions

- Fabret et. al. presents the most efficient solution to the problem so far.
- Efficiency achieved using clusters, workload adaptive algorithm and prefetching.
- Practical implementation exists.