On the Database/Network Interface in Large-Scale Publish/Subscribe Systems

slides by: Badrish Chandramouli
Duke University

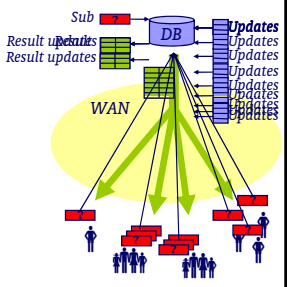
Presented by: Risi Thonangi
Instructor: Jun Yang (CPS399.28)

DUKE *Systems & Architecture*

Contents

Pub/Sub Systems

- Publish/subscribe systems
  - Many subscriptions over an input update stream (events)
  - Uses a push model which ensures timely update delivery
  - Many applications: personal, financial, security, military
- Scalability challenges
  - Too many subscriptions
  - More complex subscriptions
  - Results needed all over the network
- Two components:
  - Subscription processing
  - Notification dissemination

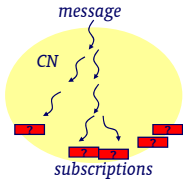


Traditional DB-centric Approach

- Traditional DB-centric approach
  - Focused on subscription processing
  - Ignored notification dissemination
- Implicit assumption: output a list of notifications, one for each affected subscription
  - $\langle Q_{i1}, msg \rangle, \langle Q_{i2}, msg \rangle, \langle Q_{i3}, msg \rangle, \ldots$
  - Potentially a *very* long list
  - Sending them to subscribers one at a time (unicast) can overwhelm the server and its outgoing network links

Network-centric approach

- Content-based networking (CN): supports message-based filter subscriptions directly in network
  - Message: $\langle attr_1{:}val_1, attr_2{:}val_2, attr_3{:}val_3, \ldots \rangle$
  - Subscription: "$attr_1$ = 'foo' and $attr_2 \in$ [low,high] and …"
- ☞ Doesn't support stateful subscriptions



Stateful subscription example

- Range-min subscription
  - Q: select MIN(PER) from STOCK where RISK between 20 and 40
- Update message ⟨SYM:foo, RISK:35, PER:25 → 20⟩
- ☞ Stateful: cannot determine its effect on Q just by looking at the message itself
  - Is there another stock in RISK range with PER < 20?

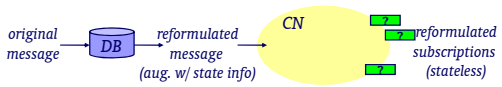
Supporting stateful subscriptions

- Just stick the DB-centric approach and a network together?
  - "List of affected subscriptions" leads to unicast
  - Multicast: map the list to group(s) first, then send
    - ☞Too many possible subsets! What groups to form?

Supporting stateful subscriptions

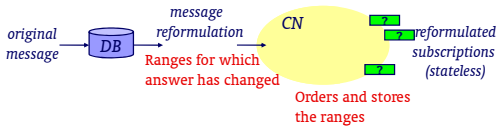
- Content-based network?
  - Naïve method: "relax" subscription into a stateless one
    - select MIN(PER) from STOCK where RISK between 20 and 40
    - ☞select PER from STOCK where RISK between 20 and 40
  - ☞Too many unnecessary notifications!
- Push state support into network of smart brokers?
  - Network controls dissemination using state
  - Complicates system design and deployment
    - Pushes complex routing logic into network
    - Lacks a clean interface between database and network
- Alternative: hybrid approach using message and subscription reformulation (next)

Server with Content-based Network (S-CN)



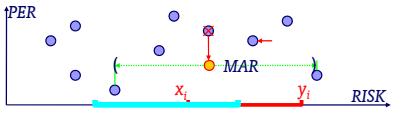
- Reformulate messages to add state info
- Reformulate subscriptions into stateless ones over new message format
- Naïve: put entire database state into message!
- Optimization problem: what's the minimal amount of info to embed?

Basic Idea of S-CN



Range-min revisited

- Q_i: MIN(PER), where RISK between x_i and y_i
- Update ⟨SYM:foo, RISK:35, PER:25 → 20⟩



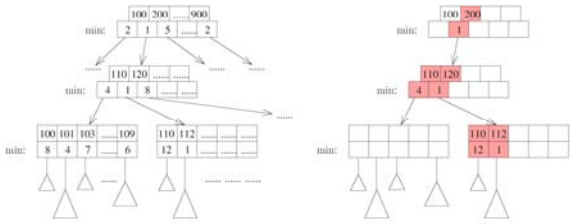
- Maximum Affected Range (MAR): extends left & right until a lower PER is encountered
- What info should DB server send out to the CN
  - Affected ⇔ RISK of update ∈ $[x_i, y_i]$ ⊆ MAR of update

Reformulation for range-min

- Message reformulation (at runtime):
  - ⟨SYM:foo, RISK:35, PER:25 → 20⟩
  - Say MAR is (17, 52)
  - ☞⟨NewMinPER:20, RISK:35, MARLeftRISK:17, MARRightRISK:52⟩
- Subscription reformulation (at registration time)
  - Q_i: MIN(PER), where RISK between x_i and y_i
  - ☞Q_i': NewMinPER, where MARLeftRisk $< x_i \leq$ RISK and RISK $\leq y_i <$ MARRightRisk
- ☞ Changing role of DB
  - From producing the *set of affected subscriptions*
  - To producing a *semantic description of the set*

Computing MAR

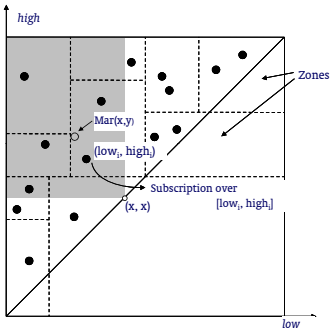
- We propose *A2B-tree*
  - Upper tier is B-tree on range attribute, lower tier is B-tree on aggregate attribute
  - Insert, lookup, update, compute MAR: $O(\log_B N)$ I/Os



Disseminating reformulated messages

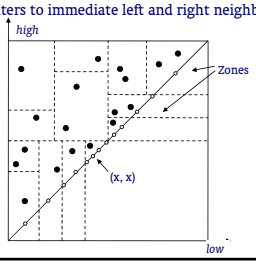
- Range-min: we use CAN (Meghdoot)
  - Minimal modification necessary
  - Can use traditional content-based networks as well

Disseminating reformulated messages



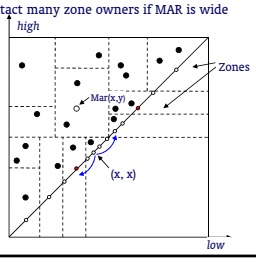
Distributing the server

- Replace central server with multiple servers
  - Maintain the database in distributed manner
  - Store data closer to subscriptions that are likely to be affected
- Map a stock with RISK=x, to point (x,x) on diagonal of CAN
  - Zone owner is responsible for all stocks within a zone
  - Maintains pointers to immediate left and right neighbors



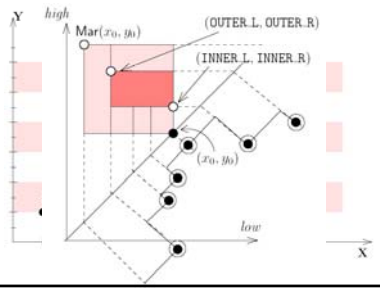
Distributing the server

- When update comes, use linear distributed traversals (to the left and right) to examine all stocks in MAR
- Advantages:
  - No bottleneck of central server
  - Underlying network can ensure load balancing
- Disadvantage:
  - May need to contact many zone owners if MAR is wide



On handling bad updates

- Bad updates are more complicated
  - An update may expose more than one new minimum
  - Each exposed new minimum generates a reformulated message



Other subscription types

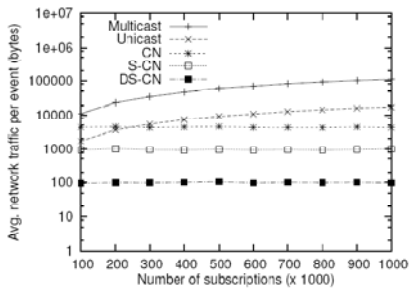
- Range-max
- Range-count/sum/average
- Range-DISTINCT
- Select-join
- Range aggregation in higher dimensions

Experimental setup

- Evaluation metrics
  - Number of overlay and IP message hops
  - Network traffic
  - Maximum node stress
  - Server-side processing time
- Workload
  - Subscriptions:
    - Synthetic 1-d range MIN, model *hot ranges*
  - Updates:
    - Synthetic: Uses a random walk model with spikes
    - Real: Stock data from Yahoo! Finance
- Setup
  - Detailed link-level simulation of 20,000 node INET topology
  - 1000 nodes participate in an overlay network

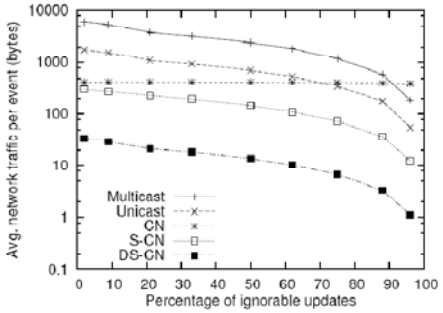
Subset of experiments

- Increasing number of subscriptions



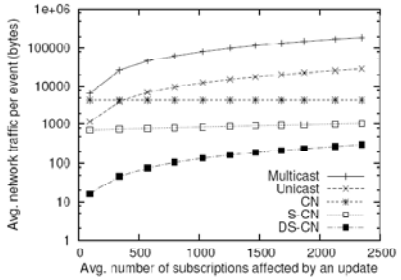
Subset of experiments

- Increasing percentage of ignorable updates

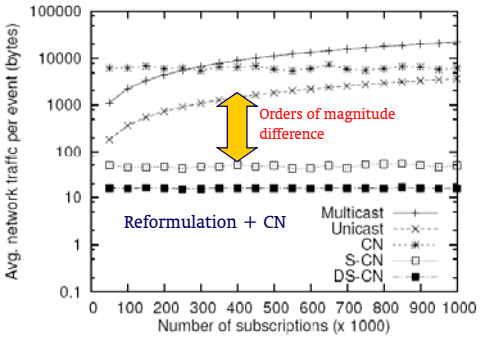


Subset of experiments

- Increasing 'average number of subscriptions affected by an update'



Experiments on real workload



- Yahoo! Stock updates +synthetic subscriptions

Discussion

- Deals with range-predicate
  - Not clear how to extend to other operators
- How scalable are the data-structures/algorithms as number of dimensions increase?
- Robustness concerns in CN
- Load-balancing in CN

Thanks!