CS 347: Parallel and Distributed Data Management

Notes 12: Publish/Subscribe
Point to Point Communication

To: Joe
From: Sally
Body: B
Publish/Subscribe Communication

Publication
Description: D
Body: B

Subscription
Query: Q
Id: I
Publish/Subscribe Communication

Publication
Description: D
Body: B

Subscription
Query: Q
Id: I

Publish/Subscribe System
Applications

• End user notifications
• Application integration
• Asynchronous indexing
• Cache invalidation
• Social networking
• ...

P/S Semantics

- **Subscribe** (query \(Q\); id \(I\)):
  
  \[
  \text{add } [Q, I] \text{ to SDB}
  \]

- **Publish** (description \(D\); body \(B\)):
  
  \[
  \text{for } [Q,I] \text{ in SDB do}
  \]
  
  \[
  \text{if } Q[D] \text{ then notify}(I, B)
  \]
P/S Semantics

- Subscribe(query Q; id I):
  add [Q, I] to SDB

- Publish(description D; body B):
  for [Q,I] in SDB do
    if Q[D] then notify(I, B)

Notify can send email, or can put B in DB(I)
(which I checks periodically)
P/Q Semantics (Alternative)

- Publish(description D; body B):
  add \([D, B]\) to PDB
- Query(query Q; id I):
  for \([D, B]\) in PDB do
    if Q[D] then notify(I, B)
P/Q Semantics (Alternative)

- Publish(description D; body B):
  add [D, B] to PDB

- Query(query Q; id I):
  for [D, B] in PDB do
    if Q[D] then notify(I, B)

Notify could just send out new material (not seen by previous queries)
P/S Features

- **Space decoupling:** interacting parties do not need to know each other
- **Time decoupling:** interacting parties do not need to actively participate at the same time
- **Synchronization decoupling:** publishers and subscribers do not block for each other
Other Communication Models

- Message passing
- RPC (remote procedure call)
- Shared Spaces (bulletin board)
- Message Queues
Description/Query Models

• (flat) Topics
  – e.g.: Topics = \{sports, business, politics, \ldots\}
Description/Query Models

• (flat) Topics
  – $T$ is set of possible topics
  – description $D$ is a subset of $T$
  – query $Q$ is a subset of $T$
  – $D$ matches $Q$ if $D \cap Q \neq \emptyset$ (empty set)
Description/Query Models

• Topic Hierarchy

- all
  - sports
    - soccer
    - football
    - rugby
  - business
    - tech
    - service
  - politics
- college
- NFL
Description/Query Models

• Topic Hierarchy

P1.D = all.sports.football
P2.D = all.sports
P3.D = {all.business.tech, all.politics}

S1.Q = all.sports.rugby
S2.Q = {all.politics.calif, all.sports.soccer}
Description/Query Models

• (flat) Topics
  – $T$ is tree of topics (or DAG?)
  – description $D$ is a set of paths in $T$
  – query $Q$ is a set of paths in $T$
  – description path $d$ matches query path $q$
    if $q$ is a prefix of $d$
  – $D$ matches $Q$ if there exists a path in $Q$
    that matches a path in $D$
Description/Query Models

• Key-value pairs

\[
\begin{align*}
P1.D &= \{[\text{price, 50}], [\text{size, L}]\} \\
P2.D &= \{[\text{price, 80}]\} \\
P3.D &= \{[\text{size, M}], [\text{size, L}]\}
\end{align*}
\]

\[
\begin{align*}
S1.Q &= \{[\text{price, 50}]\} \\
S2.Q &= \{[\text{price, 50}], [\text{size, M}]\}
\end{align*}
\]
Description/Query Models

• Key-value pairs

P1.D={\{price, 50\}, \{size, L\}}
P2.D={\{price, 80\}}
P3.D={\{size, M\}, \{size, L\}}

S1.Q={\{price, 50\}}
S2.Q={\{price, 50\}, \{size, M\}}

s3.Q= [price > 40] AND [size = L]
Matching Descriptions to Queries

Publication
Description: D
Body: B

SDB

Subscription
Query: Q
Id: I
Matching Descriptions to Queries

Publication
Description: D
Body: B

Subscription
Query: Q
Id: I
Generic Distributed Matching

Pi (to any row)

Sj (to any column)

Pi/Sj match
Generic Distributed Matching

- pub to one of \{a,b\}, \{c,d\}, \{e,f\}
- sub to one of \{a,c,e\}, \{b,d,f\}

sound familiar??
Can use any abort/commit quorums

pub to one of \{\{a,b\}, \{c,d\}\}
sub to one of \{\{a,c\}, \{a,d\}, \{b,c\}, \{b,d\}\}
Generic Distributed Matching

- pub to one of \{a,b\}, \{c,d\}, \{e,f\}
- sub to one of \{a,c,e\}, \{b,d,f\}

Issues:
- Replicated data (subs)
- Load balance
- Total work
Simple Cost Model

- At node with x subs, y pubs:
  work(x,y), data(x)
- Ex: work(x,y)=xy (case (i))
  work(x,y)=y  (case (ii))
  data(x)=x
- For 6 node grid: s subs, p pubs
- Each node: s/2 subs, p/3 pubs
- Scenario I: total data = 6*data(s/2) = 3s
  total work = 6*work(s/2, p/3) = 6(p/3)= 2p (case (ii))
- Scenario II: total data = 3s (same as before)
  total work = 6*work(s/2, p/3) = 6(s/2)(p/3) = sp (case (i))
- Compare to single node scenario: total data = s
  total work = p (case (ii)) or sp (case (i))
Topic Distributed Matching

Say \( T = \{ t_1, t_2, t_3 \} \)

\[ S_j \text{ (to one topic)} \]

\[ \text{(to one topic)} \]
Say $T = \{t_1, t_2, t_3\}$

Sound familiar?
- Data fragmentation (in this case subs)
- Query localization (in this case pubs)
Matching with Topic Hierarchy

Publication dissemination tree:
Matching with Topic Hierarchy

Publication dissemination tree:
NetNews worked like this

Publication dissemination tree:

- t
  - t/1
    - t/1/1
    - t/1/2
  - t/2
  - t/3
    - t/3/1
    - t/3/2
    - t/3/3

note replication
Discussion: Twitter

follows

S(e) = \{a,d\}
pubs by e have description “e”
body of pub is 140 char max
users periodically check for notifications
Discussion: Twitter

follows inv lists:
S(a): b
S(b): a,c,d,e
S(c): d
S(d): -
S(e): a, d

is-followed inv lists:
S^{-1}(a): b,e
S^{-1}(b): a
S^{-1}(c): b
S^{-1}(d): b,c,e
S^{-1}(e): b
Strawman Architecture - Centralized

users

front end

back end

follows inv lists

is-followed inv lists

log (all pubs)

pubs by user

notify by user

a

b

c

d

...
Strawman Architecture - Centralized

Question: Advantages/disadvantages of "notify by user" storage? Of "pubs by user"?

<table>
<thead>
<tr>
<th>users</th>
</tr>
</thead>
<tbody>
<tr>
<td>front end</td>
</tr>
<tr>
<td>back end</td>
</tr>
</tbody>
</table>

follows inv lists

is-followed inv lists

log (all pubs)

pubs by user

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>. . .</th>
</tr>
</thead>
</table>

notify by user

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>. . .</th>
</tr>
</thead>
</table>
Strawman Architecture- Distributed

How do we split back end???
Strawman Architecture- Distributed

notify by user
a

notify by user
b

c

notify by user
c

notify by user
d

pubs by user
a

pubs by user
b

c

pubs by user
c

pubs by user
d

follows inv lists
back end 1

follows inv lists
back end 2

is-followed inv lists
log (1/2 pubs)

is-followed inv lists
log (1/2 pubs)
Dynamic Dissemination Tree

Publication dissemination tree:
Dynamic Dissemination Tree

Publication dissemination tree:

- t
- t/1
- t/2
- t/3
- t/1/1
- t/1/2
- t/3/1
- t/3/2
- t/3/3
- all pubs
- t/1,2
- t/3
- new node interest: t/1
- wireless range

CS347
Dynamic Dissemination Tree

Publication dissemination tree:
Dynamic Dissemination Tree

Publication dissemination tree:
Dynamic Dissemination Tree

![Diagram of a dynamic dissemination tree with nodes t, t/1, t/2, t/3, t/1/1, t/1/2, t/3/1, t/3/2, t/3/3, and arrows indicating publication dissemination tree branches.]

dissemination nodes can also publish:

**Publication dissemination tree:**

- All pubs
- t/1,2
- t/3
- t/1/1
- t/3/2

new pub 1: t/1/2
new pub 2: t/3/2

new node
interest: t/1/1
t/3/2
Matching at One Node

- Set \{ [Q_j, I_j] \} of stored subscriptions
- Match one publication \( p \) from stream

\[ p[D,B] \rightarrow \{ [Q_j, I_j] \} \]

- Index of queries \( Q_j \) different from index of descriptions (content)
Example

Subscriptions:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>(a,b)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s2</td>
<td>(a,d)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s3</td>
<td>(a,d,e)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4</td>
<td>(b,f)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s5</td>
<td>(c,d,e,f)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

sample publication

| a | c | a | f | b | c |

Inverted Lists:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>s1</td>
<td>s2</td>
</tr>
<tr>
<td>b</td>
<td>s1</td>
<td>s4</td>
</tr>
<tr>
<td>c</td>
<td>s5</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>s2</td>
<td>s3</td>
</tr>
<tr>
<td>e</td>
<td>s3</td>
<td>s5</td>
</tr>
<tr>
<td>f</td>
<td>s4</td>
<td>s5</td>
</tr>
</tbody>
</table>

- Match semantics for example: s1 = (a,b) matches pub if both a and b appear in pub
- Can generalize, e.g., sj = a \(\land (b \lor c)\)
  handle as two subs, sj1 = (a \(\land b\)), sj2 = (a \(\land c\))
### Example

**Subscriptions:**

- **s1**: (a,b)
- **s2**: (a,d)
- **s3**: (a,d,e)
- **s4**: (b,f)
- **s5**: (c,d,e,f)

#### Inverted Lists:

<table>
<thead>
<tr>
<th></th>
<th>s1</th>
<th>s2</th>
<th>s3</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
<td>s5</td>
</tr>
<tr>
<td>d</td>
<td>s2</td>
<td>s3</td>
<td>s5</td>
</tr>
<tr>
<td>e</td>
<td></td>
<td></td>
<td>s5</td>
</tr>
<tr>
<td>f</td>
<td>s4</td>
<td>s5</td>
<td></td>
</tr>
</tbody>
</table>

**sample publication**

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>c</td>
<td>a</td>
<td>f</td>
<td>b</td>
<td>c</td>
</tr>
</tbody>
</table>

- Intersection of lists (|=null) not useful
- Union of lists \{s1,s2,s3,s4,s5\} gives candidate subscriptions
- Need to check each candidate (e.g., s1 matches, s2 does not)
Counting Method

Subscriptions:

<table>
<thead>
<tr>
<th></th>
<th>total</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>s2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>s3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>s4</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>s5</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

distinct word set

a b c f

Inverted Lists:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>s1</td>
<td>s2</td>
</tr>
<tr>
<td>b</td>
<td>s1</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>s2</td>
<td>s3</td>
</tr>
<tr>
<td>e</td>
<td>s3</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td></td>
<td>s4</td>
</tr>
</tbody>
</table>

sample publication

a c a f b c
Counting Method

Subscriptions:

<table>
<thead>
<tr>
<th>s1</th>
<th>(a,b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>s2</td>
<td>(a,d)</td>
</tr>
<tr>
<td>s3</td>
<td>(a,d,e)</td>
</tr>
<tr>
<td>s4</td>
<td>(b,f)</td>
</tr>
<tr>
<td>s5</td>
<td>(c,d,e,f)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>total</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>s2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>s3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>s4</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>s5</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

Distinct word set

| a | b | c | f |

Inverted Lists:

<table>
<thead>
<tr>
<th></th>
<th>s1</th>
<th>s2</th>
<th>s3</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>s1</td>
<td></td>
<td>s4</td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
<td>s2</td>
<td>s3</td>
</tr>
<tr>
<td>e</td>
<td>s3</td>
<td>s5</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td></td>
<td>s4</td>
<td>s5</td>
</tr>
</tbody>
</table>

Sample publication

a c a f b c

- As we union lists, count number of times each sub appears
- If count $\geq$ total, then sub matches
### Key Method

#### Subscriptions:

<table>
<thead>
<tr>
<th>s1</th>
<th>(a,b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>s2</td>
<td>(a,d)</td>
</tr>
<tr>
<td>s3</td>
<td>(a,d,e)</td>
</tr>
<tr>
<td>s4</td>
<td>(b,f)</td>
</tr>
<tr>
<td>s5</td>
<td>(c,d,e,f)</td>
</tr>
</tbody>
</table>

#### Inverted Lists:

<table>
<thead>
<tr>
<th></th>
<th>s1 [1, (b)]</th>
<th>s2 [1, (d)]</th>
<th>s3 [2, (d,e)]</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
<td>s5 [3, (d,e,f)]</td>
</tr>
<tr>
<td>d</td>
<td></td>
<td>null</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td></td>
<td>null</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td></td>
<td>null</td>
<td></td>
</tr>
</tbody>
</table>

#### Sample Publication

| a | c | a | f | b | c |

#### Distinct Word Set

| a | b | c | f |
Key Method

Subscriptions:

- **s1**: (a,b)
- **s2**: (a,d)
- **s3**: (a,d,e)
- **s4**: (b,f)
- **s5**: (c,d,e,f)

Sample publication:

```
  a  c  a  f  b  c
```

Distinct word set:

```
a  b  c  f
```

Inverted Lists:

<table>
<thead>
<tr>
<th></th>
<th>s1 [1, (b)]</th>
<th>s2 [1, (d)]</th>
<th>s3 [2, (d,e)]</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
<td></td>
<td>null</td>
</tr>
<tr>
<td>e</td>
<td></td>
<td></td>
<td>null</td>
</tr>
<tr>
<td>f</td>
<td></td>
<td></td>
<td>null</td>
</tr>
</tbody>
</table>

- sub in only one inverted list
- each IL entry contains other terms in sub
- occurrence table is for fast hash lookup
Summary: Publish/Subscribe

- P/S semantics
- Various query/description models
- Distributed matching
- Matching at one node