Notes 10: Publish/Subscribe Systems
Point-to-Point Communication

To: Alice
From: Bob
Message: <M>
Publish/Subscribe Communication

<table>
<thead>
<tr>
<th>Publication</th>
<th>Subscription</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description: &lt;D&gt;</td>
<td>Query: &lt;Q&gt;</td>
</tr>
<tr>
<td>Message: &lt;M&gt;</td>
<td>Identifier: &lt;I&gt;</td>
</tr>
</tbody>
</table>
Publish/Subscribe Communication

- **Description:** 
- **Message:** 

<table>
<thead>
<tr>
<th>Publication</th>
<th>Subscription</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description: &lt;D&gt;</td>
<td></td>
</tr>
<tr>
<td>Message: &lt;M&gt;</td>
<td></td>
</tr>
</tbody>
</table>

- **Query:** <Q>
- **Identifier:** <I>
Publish/Subscribe Applications

Downstream/end user notifications
Operational monitoring
Log aggregation
Application integration
Stream processing
Social networking
Publish/Subscribe Semantics

subscribe(Q, I):
    add [Q, I] to SDB // subscription database
update(D, M):
    for [Q, I] in SDB do
        if match(D, Q) then notify(I, M)
Publish/Subscribe Semantics

subscribe(Q, I):
    add [Q, I] to SDB
update(D, M):
    for [Q, I] in SDB do
        if match(D, Q) then notify(I, M)

Can send email, make remote procedure call, write M to DBI, etc.

<table>
<thead>
<tr>
<th>Publication</th>
<th>Subscription</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description: &lt;D&gt;</td>
<td>Query: &lt;Q&gt;</td>
</tr>
<tr>
<td>Message: &lt;M&gt;</td>
<td>Identifier: &lt;I&gt;</td>
</tr>
</tbody>
</table>
Publish/Subscribe Semantics

publish(D, M):
    add [D, M] to PDB  // publication database
query(Q, I):
    for [D, M] in PDB do
        if match(D, Q) then notify(I, M)
Publish/Subscribe 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 (through channels)
Message queues
Remote procedure calls (RPCs)
Shared memory (bulletin boards)
Description/Query Models

Flat topics
E.g., Topics = \{business, politics, sports, ...\}

Publication $P_1$
- Description: \{s\}
- Message: ...

Publication $P_2$
- Description: \{b, s\}
- Message: ...

Subscription $S_1$
- Query: \{b\}
- Identifier: ...

Subscription $S_2$
- Query: \{p, s\}
- Identifier: ...

SDB
Description/Query Models

Flat topics
E.g., Topics = \{business, politics, sports, ...\}

T is set of possible topics
Description \( D \) is a subset of \( T \)
Query \( Q \) is a subset of \( T \)

\[
\text{match}(D, Q) = \text{true if } D \cap Q \neq \emptyset
\]
Description/Query Models

Topic hierarchy

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

Topic hierarchy

$P_1.\mathcal{D} = \{ \text{all/sports/football} \}$
$P_2.\mathcal{D} = \{ \text{all/sports} \}$
$P_3.\mathcal{D} = \{ \text{all/business/tech, all/politics} \}$

$S_1.\mathcal{Q} = \{ \text{all/sports/soccer} \}$
$S_2.\mathcal{Q} = \{ \text{all/politics/canada, all/sports/hockey} \}$
Description/Query Models

Topic hierarchy

T is tree of topics (or DAG?)
Description $\mathcal{D}$ is a set of paths in $T$
Query $\mathcal{Q}$ is a set of paths in $T$

Description path $d$ matches query path $q$ if $q$ is a prefix of $d$
$\mathcal{D}$ matches $\mathcal{Q}$ if there exists a path in $\mathcal{Q}$ that matches a path in $\mathcal{D}$
Description/Query Models

Key-value pairs

\[ P_1.D = \{ \text{[price, 50], [size, L]} \} \]
\[ P_2.D = \{ \text{[price, 80]} \} \]
\[ P_3.D = \{ \text{[size, M], [size, L]} \} \]

\[ S_1.Q = \{ \text{[price, 50]} \} \]
\[ S_2.Q = \{ \text{[price, 50], [size, M]} \} \]
Description/Query Models

Key-value pairs

\[ P_1.D = \{ [\text{price, 50}], [\text{size, L}] \} \]
\[ P_2.D = \{ [\text{price, 80}] \} \]
\[ P_3.D = \{ [\text{size, M}], [\text{size, L}] \} \]

\[ S_1.Q = \{ [\text{price, 50}] \} \]
\[ S_2.Q = \{ [\text{price, 50}], [\text{size, M}] \} \]
\[ S_3.Q = [\text{price > 40}] \text{ AND } [\text{size \neq L}] \]
Matching Description to Queries

<table>
<thead>
<tr>
<th>Publication</th>
<th>SDB</th>
<th>Subscription</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>&lt;D&gt;</td>
<td>Query:</td>
</tr>
<tr>
<td>Message:</td>
<td>&lt;M&gt;</td>
<td>Identifier:</td>
</tr>
</tbody>
</table>

SDB
Matching Description to Queries

Publication

Description: <D>
Message: <M>

Subscription

Query: <Q>
Identifier: <I>
Generic Distributed Matching

\( P_i \) (To any row)

\( S_j \) (To any column)

match \( (Q_j, D_i) \)
Generic Distributed Matching

Publish to one of \{ a, b \}, \{ c, d \}, \{ e, f \}
Subscribe to one of \{ a, c, e \}, \{ b, d, f \}
Generic Distributed Matching

Publish to one of \{ a, b \}, \{ c, d \}, \{ e, f \}
Subscribe to one of \{ a, c, e \}, \{ b, d, f \}
Generic Distributed Matching

Publish to one of \{ a, b \}, \{ c, d \}, \{ e, f \}
Subscribe to one of \{ a, c, e \}, \{ b, d, f \}

Can use any quorum
Generic Distributed Matching

Publish to one of \{ a, b \}, \{ c, d \}
Subscribe to one of \{ a, c \}, \{ a, d \}, \{ b, c \}, \{ b, d \}
Generic Distributed Matching

Cost
Replicated data (stored subscriptions)
Balanced load (processed publications)
Generic Distributed Matching

Cost example
At node with $x$ subscriptions handling $y$ publications

data($x$) = $x$

Scenario 1: $\text{work}(x, y) = xy$
Scenario 2: $\text{work}(x, y) = y$
Generic Distributed Matching

Cost example
For 6-node grid with $s$ subscriptions and $p$ publications
Each node handles $s/2$ subscriptions, $p/3$ publications

Scenario 1
$\text{total\_data} = 6 \text{ data}(s/2) = 3s$
$\text{total\_work} = 6 \text{ work}(s/2, p/3) = 6 (s/2) \times (p/3) = sp$

Scenario 2
$\text{total\_data} = 6 \text{ data}(s/2) = 3s$
$\text{total\_work} = 6 \text{ work}(s/2, p/3) = 6 (p/3) = 2p$
Generic Distributed Matching

Cost example
For a single node

Scenario 1
total_data = s
total_work = sp

Scenario 2
total_data = s
total_work = p
Topic Matching

E.g., $T = \{ t_1, t_2, t_3 \}$
Topic Matching

E.g., $T = \{ t_1, t_2, t_3 \}$
Topic Matching

E.g., $T = \{ t_1, t_2, t_3 \}$

Data fragmentation
E.g., on subscriptions
Query localization
E.g., for publications
Topic Hierarchy Matching

Publication dissemination tree

All publications $\rightarrow$ $t/\{1,2\}$

$\rightarrow$ $t/3$

$\rightarrow$ $t/3/1$

$t/3/\{2,3\}$

$t/3/1$ publications processed here
Topic Hierarchy Matching

Publication dissemination tree

All publications

Subscriptions to t/3 or (t/3/1 or t/3/2)

Subscriptions to t/3/1 publications processed here

Subscriptions to t/3/2 or (t/3/2 or t/3/3)
Topic Hierarchy Matching

Publication dissemination tree

Subscriptions to \( t/3 \) or (\( t/3/1 \) or \( t/3/2 \))

Subscriptions to \( t/3/2 \) or (\( t/3/2 \) or \( t/3/3 \))

Note the replication
Dynamic Dissemination Tree

Publication dissemination tree

All publications

New node
Subscriptions: \{t/1\}

Neighborhood
Dynamic Dissemination Tree

All publications → t/1, t/2, t/3

Subscriptions: {t/1}

Neighborhood

New node

Subscriptions: {t/1}
Dynamic Dissemination Tree

Publication dissemination tree

All publications → t\{1,2\} → t/1 → t/2 → t/3

Subscriptions: {t/1/1, t/3/2}

New node

Subscriptions: {t/1/1, t/3/2}

Neighborhood
Dynamic Dissemination Tree

Publication dissemination tree

Neighborhood

Subscriptions: \{t/1/1, t/3/2\}
Dynamic Dissemination Tree

Publication dissemination tree

What if all nodes can publish?

New publication t/1/2
New publication: t/3/2
Matching at One Node

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

Match semantics
Each publication \( p \) is bag of terms
Each subscription \( s \) has set of terms
There is a match when all \( s \) terms appear in \( p \)
Matching at One Node

Example

<table>
<thead>
<tr>
<th>Subscriptions</th>
<th>Inverted lists</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_1 = { a, b } )</td>
<td>a ( \rightarrow ) ( s_1 ) ( s_2 ) ( s_3 )</td>
</tr>
<tr>
<td>( s_2 = { a, d } )</td>
<td>b ( \rightarrow ) ( s_1 ) ( s_4 )</td>
</tr>
<tr>
<td>( s_3 = { a, d, e } )</td>
<td>c ( \rightarrow ) ( s_5 )</td>
</tr>
<tr>
<td>( s_4 = { b, f } )</td>
<td>d ( \rightarrow ) ( s_2 ) ( s_3 ) ( s_5 )</td>
</tr>
<tr>
<td>( s_5 = { c, d, e, f } )</td>
<td>e ( \rightarrow ) ( s_3 ) ( s_5 )</td>
</tr>
<tr>
<td>( f \rightarrow )</td>
<td>( s_4 ) ( s_5 )</td>
</tr>
</tbody>
</table>

Sample publication

\( a \ c \ a \ f \ b \ c \)

\( s_1 = \{ a, b \} \) matches publication if both terms \( a \) and \( b \) appear in it

Can generalize, e.g., \( s_j = a \land (b \lor c) \) handled as two subscriptions: \( s_{j_1} = (a \land b) \), \( s_{j_2} = (a \land c) \)
Matching at One Node

Example

**Subscriptions**

- $s_1 \{a, b\}$
- $s_2 \{a, d\}$
- $s_3 \{a, d, e\}$
- $s_4 \{b, f\}$
- $s_5 \{c, d, e, f\}$

**Inverted lists**

```
<table>
<thead>
<tr>
<th>a</th>
<th>s_1</th>
<th>s_2</th>
<th>s_3</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>s_1</td>
<td></td>
<td>s_4</td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
<td>s_5</td>
</tr>
<tr>
<td>d</td>
<td>s_2</td>
<td>s_3</td>
<td>s_5</td>
</tr>
<tr>
<td>e</td>
<td>s_3</td>
<td>s_5</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>s_4</td>
<td>s_5</td>
<td></td>
</tr>
</tbody>
</table>
```

**Sample publication**

- $a \ c \ a \ f \ b \ c$

Intersection of lists for $a, b, c, f$ not useful ($= \emptyset$)

Union of lists $= \{ s_1, s_2, s_3, s_4, s_5 \}$ gives candidate subscriptions

Need to check each candidate (e.g., $s_1$ matches but $s_2$ does not)
Matching at One Node

Counting method

**Subscriptions**

- \( s_1 \) \{ a, b \}
- \( s_2 \) \{ a, d \}
- \( s_3 \) \{ a, d, e \}
- \( s_4 \) \{ b, f \}
- \( s_5 \) \{ c, d, e, f \}

**Sample publication**

- \( a \ c \ a \ f \ b \ c \)

**Inverted lists**

- \( a \rightarrow s_1 \ s_2 \ s_3 \)
- \( b \rightarrow s_1 \ s_4 \)
- \( c \rightarrow s_5 \)
- \( d \rightarrow s_2 \ s_3 \ s_5 \)
- \( e \rightarrow s_3 \ s_5 \)
- \( f \rightarrow s_4 \ s_5 \)

**Total**

<table>
<thead>
<tr>
<th></th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_1 )</td>
<td>2</td>
</tr>
<tr>
<td>( s_2 )</td>
<td>2</td>
</tr>
<tr>
<td>( s_3 )</td>
<td>3</td>
</tr>
<tr>
<td>( s_4 )</td>
<td>2</td>
</tr>
<tr>
<td>( s_5 )</td>
<td>4</td>
</tr>
</tbody>
</table>

**Distinct term set**

- \( a \ b \ c \ f \)
Matching at One Node

Counting method

Subscriptions

- $s_1 \{ a, b \}$
- $s_2 \{ a, d \}$
- $s_3 \{ a, d, e \}$
- $s_4 \{ b, f \}$
- $s_5 \{ c, d, e, f \}$

Inverted lists

- $a \rightarrow s_1 | s_2 | s_3$
- $b \rightarrow s_1 | s_4$
- $c \rightarrow s_5$
- $d \rightarrow s_2 | s_3 | s_5$
- $e \rightarrow s_3 | s_5$
- $f \rightarrow s_4 | s_5$

Sample publication

- $a \ c \ a \ f \ b \ c$

Distinct term set

- $a \ b \ c \ f$

When computing the union, count number of times each subscription appears
If count $\geq$ total then subscription matches

<table>
<thead>
<tr>
<th>Total</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>2</td>
</tr>
<tr>
<td>$s_2$</td>
<td>2</td>
</tr>
<tr>
<td>$s_3$</td>
<td>3</td>
</tr>
<tr>
<td>$s_4$</td>
<td>2</td>
</tr>
<tr>
<td>$s_5$</td>
<td>4</td>
</tr>
</tbody>
</table>
Matching at One Node

Counting method

Subscriptions

- $s_1 \{ a, b \}$
- $s_2 \{ a, d \}$
- $s_3 \{ a, d, e \}$
- $s_4 \{ b, f \}$
- $s_5 \{ c, d, e, f \}$

Inverted lists

- $a \rightarrow s_1 \ s_2 \ s_3$
- $b \rightarrow s_1 \ s_4$
- $c \rightarrow s_5$
- $d \rightarrow s_2 \ s_3 \ s_5$
- $e \rightarrow s_3 \ s_5$
- $f \rightarrow s_4 \ s_5$

Sample publication

- $a\ c\ a\ f\ b\ c$

Distinct term set

- $a\ b\ c\ f$

When computing the union, count number of times each subscription appears

If count $\geq$ total then subscription matches
Matching at One Node

Counting method

<table>
<thead>
<tr>
<th>Subscriptions</th>
<th>Inverted lists</th>
<th>Total</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>s₁ { a, b }</td>
<td>a → s₁ s₂ s₃</td>
<td>S₁</td>
<td>2</td>
</tr>
<tr>
<td>s₂ { a, d }</td>
<td>b → s₁ s₄</td>
<td>S₂</td>
<td>2</td>
</tr>
<tr>
<td>s₃ { a, d, e }</td>
<td>c → s₅</td>
<td>S₃</td>
<td>3</td>
</tr>
<tr>
<td>s₄ { b, f }</td>
<td>d → s₂ s₃ s₅</td>
<td>S₄</td>
<td>2</td>
</tr>
<tr>
<td>s₅ { c, d, e, f}</td>
<td>e → s₃ s₅</td>
<td>S₅</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>f → s₄ s₅</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Sample publication
a c a f b c

Distinct term set
a b c f

When computing the union, count number of times each subscription appears
If count ≥ total then subscription matches
Matching at One Node

Counting method

Subscriptions

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>s₁</td>
<td>{ a, b }</td>
</tr>
<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

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

Sample publication

a c a f b c

Counting

<table>
<thead>
<tr>
<th>Subscriptions</th>
<th>Total</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>s₁</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>s₂</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>s₃</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>s₄</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>s₅</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

Distinct term set

| a b c f |

When computing the union, count number of times each subscription appears
If count \( \geq \) total then subscription matches
Matching at One Node

Counting method

<table>
<thead>
<tr>
<th>Subscriptions</th>
<th>Inverted lists</th>
</tr>
</thead>
<tbody>
<tr>
<td>s₁  { a, b }</td>
<td>a → s₁ s₂ s₃</td>
</tr>
<tr>
<td>s₂  { a, d }</td>
<td>b → s₁ s₄</td>
</tr>
<tr>
<td>s₃  { a, d, e }</td>
<td>c → s₅</td>
</tr>
<tr>
<td>s₄  { b, f }</td>
<td>d → s₂ s₃ s₅</td>
</tr>
<tr>
<td>s₅  { c, d, e, f }</td>
<td>e → s₃ s₅</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>2</td>
</tr>
<tr>
<td>S₂</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S₃</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>S₄</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>S₅</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

Distinct term set

a b c f

Sample publication

a c a f b c

When computing the union, count number of times each subscription appears
If count \( \geq \) total then subscription matches
Matching at One Node

Key method

<table>
<thead>
<tr>
<th>Subscriptions</th>
<th>Inverted lists</th>
<th>Occurrence table</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_1 ) { a, b }</td>
<td>( a \rightarrow s_1[1, {b}] )</td>
<td>a</td>
</tr>
<tr>
<td>( s_2 ) { a, d }</td>
<td>( b \rightarrow s_4[1, {f}] )</td>
<td>b</td>
</tr>
<tr>
<td>( s_3 ) { a, d, e }</td>
<td>( c \rightarrow s_5[3, {d, e, f}] )</td>
<td>c</td>
</tr>
<tr>
<td>( s_4 ) { b, f }</td>
<td>( d \rightarrow ) null</td>
<td>f</td>
</tr>
<tr>
<td>( s_5 ) { c, d, e, f }</td>
<td>( e \rightarrow ) null</td>
<td></td>
</tr>
<tr>
<td>( f \rightarrow ) null</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sample publication

a c a f b c

Distinct term set

a b c f
Matching at One Node

Key method

<table>
<thead>
<tr>
<th>Subscriptions</th>
<th>Inverted lists</th>
<th>Occurrence table</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$ { a, b }</td>
<td>$a \rightarrow s_1[1, {b}]$</td>
<td>$a$</td>
</tr>
<tr>
<td>$s_2$ { a, d }</td>
<td>$b \rightarrow s_4[1, {f}]$</td>
<td>$b$</td>
</tr>
<tr>
<td>$s_3$ { a, d, e }</td>
<td>$c \rightarrow s_5[3, {d, e, f}]$</td>
<td>$c$</td>
</tr>
<tr>
<td>$s_4$ { b, f }</td>
<td>$d \rightarrow \text{null}$</td>
<td>$f$</td>
</tr>
<tr>
<td>$s_5$ { c, d, e, f }</td>
<td>$e \rightarrow \text{null}$</td>
<td></td>
</tr>
</tbody>
</table>

Sample publication

| a c a f b c |

Each subscription only appears in one inverted list
Each inverted list entry contains the additional terms in the subscription
The occurrence table is for fast (hash) lookup for matching additional terms

Distinct term set

| a b c f |
S(e) = \{ a, d \}

Publications by e have description “e”

Messages are 140 characters max

Users periodically check for updates
Case Study: Twitter

Follows inverted lists
S(a): b
S(b): a, c, d, e
S(c): d
S(d): -
S(e): a, d

Is-followed inverted lists
S⁻¹(a): b, e
S⁻¹(b): a
S⁻¹(c): b
S⁻¹(d): b, c, e
S⁻¹(e): b
Twitter Architecture

Centralized

Users

Frontend

Backend

Log (all publications)

Follows inverted lists

Is-followed inverted lists

Publications by user

Notifications by user

a
b
c
d

a
b
c
d
Twitter Architecture

Distributed

Frontend  Frontend  Frontend

Backend

Users

Follows inverted lists
Is-followed inverted lists

Log (all publications)

Publications by user
a  b  c  d

Notifications by user
a  b  c  d
Twitter Architecture

Distributed

Users

How to split backends?

Frontend
Frontend
Frontend

Backend

Follows inverted lists

Is-followed inverted lists

Publications by user
Notifications by user

a
b
c
d

a
b
c
d

Log (all publications)
Twitter Architecture

Distributed

Notifications by user
a
... 
c

Publications by user
a
... 
c

Frontend

Backend

Follows lists

Is-followed lists

Log (1/2 publications)

Notifications by user
b
d...

Publications by user
b
d...

CS 347

Notes 10

56
Case Study: Kafka

Apache Kafka
Distributed publish-subscribe messaging system
Distributed, partitioned, replicated commit log
Kafka Publishers

Set of flat *topics*
One or more *partitions* per topic
Total immutable order and persistence of *messages* in partition
*Producers* decide which message goes to which partition

Diagram:
- Partition 1: \(m_{11}, m_{12}, m_{13}, m_{14}\)
- Partition 2: \(m_{21}, m_{22}\)
- Partition 3: \(m_{31}, m_{32}, m_{33}\)

Writes

Time
Kafka Subscribers

Subscribers organized in *consumer groups*
Within each group, each partition can be read by $\leq 1$ consumer
Kafka Scenarios

Publish/subscribe (broadcast to all subscribers)
1 consumer per group

Topic $T_1$
Partition 1
Interested in all $T_1$ updates
Partition 2

Topic $T_2$
Partition 1
Interested in some $T_1$ updates (partition 2) and all $T_2$ updates
Kafka Scenarios

Message queue (send to one in a pool of consumers)
All consumer in a single group

E.g., round robin
Kafka Implementation

Partitions act like database logs
Are truncated periodically

Partitions can be replicated using RPWP
Each server can be primary for some, backup for others
Good for load balancing across servers
Summary

Publish/subscribe semantics
Description/query models
Distributed matching
  Generic
  Topic
  Topic hierarchy
Matching at one node
Case studies